UNIVERSITY OF CATANIA
Department of Mathematics and Computer Science
A Master of Science degree in Computer Science.

Nexus Energy Distributed Service

Final Project Report

Alfio Spoto
Gabriele Ruggieri

Professor Emiliano Alessio Tramontana

Academic Year 2025 - 2026

Contents

Chapter 1

Introduction and Motivation

1.1 The Global Energy Context and Energy Transition

Over the last decade, energy resource management has shifted from being a mere operational cost item to representing a strategic challenge of global scope, influenced by sustainability policies and stringent international agreements [1]. The transition towards renewable energy sources, while necessary to achieve the decarbonization goals imposed by the European Green Deal, introduces an intrinsic variability into the power grid that traditional monitoring systems, designed for stable and predictable loads, struggle to manage efficiently [2].

Energy can no longer be understood as a static resource, but rather as a high-dimensional information flow. In this scenario, energy efficiency is no longer pursued exclusively through hardware or structural improvements (such as building envelope retrofitting), but through control software optimization and granular analysis of consumption patterns. Discrete monitoring, based on monthly or daily readings, has given way to high-frequency telemetry, where IoT (*Internet of Things*) sensors sample data with hourly or sub-hourly granularity, generating streams definable as energy Big Data [3].

1.2 Limitations of Conventional Systems and the Need for Abstraction

Classical building management systems, known as *Building Management Systems* (BMS) or legacy SCADA infrastructures, typically operate on reactive logics founded on fixed deterministic thresholds [4]. Such approaches present structural criticalities that drastically limit the resilience and intelligence of modern infrastructures:

- **Contextual Rigidity and False Alarms:** A consumption threshold considered optimal for a winter Monday morning proves completely inadequate for a summer Sunday or a corporate shutdown period. This lack of context leads to a proliferation of false positives (unjustified alarms) or, conversely, to missed detections of latent performance drifts.
- **Absence of Temporal Memory (Temporal Blindness):** Building energy consumption is a stochastic process with strong temporal dependency. Legacy ar-

chitectures often ignore this correlation, failing to consider environmental thermal inertia or HVAC system operating cycles, which make current consumption a direct function of previous states [5].

- **Univariate Analysis and Information Isolation:** Often, the alarm is triggered exclusively on the instantaneous power value (kW), ignoring crucial exogenous variables such as external temperature, humidity, or real occupancy rates. Without integrating these "features," the system is unable to distinguish between a legitimate load (e.g., heating activation on a freezing day) and waste (e.g., heating on in empty rooms).

1.3 The Digital Twin Paradigm in Predictive Monitoring

The **Nexus** project was established with the aim of overcoming the dichotomy between data collection and decisional interpretation. The core of the innovation lies in the concept of the **Digital Twin** [4]. A Digital Twin is not a simple graphical representation or a 3D CAD model, but a dynamic computational instance that evolves in real-time parallel to the physical asset.

Through the use of LSTM-type Recurrent Neural Networks (RNN), Nexus constructs a virtual baseline representing the expected "physiological" behavior of the building. This Digital Twin is queried every hour to generate a forecast: if the actual data recorded by sensors deviates from the model's projection beyond a statistical confidence corridor, the system identifies an anomaly. This approach radically transforms maintenance: operations are no longer conducted only after a failure (corrective maintenance) or at fixed intervals (preventive maintenance), but according to the paradigm of **predictive maintenance**, identifying inefficiencies invisible to classical systems [2].

1.4 Research Objectives and Scientific Contribution

This work documents the entire engineering lifecycle of the Nexus system, emphasizing implementation robustness and result validation. The research aims to demonstrate that a distributed system, based on microservices and Artificial Intelligence, can drastically reduce energy operating costs. The main contributions include:

1. **Cyclical Feature Engineering:** The application of sinusoidal transformations to model the cyclical nature of time and thermal inertia.
2. **Optimized Neural Architecture:** The design of an LSTM model implemented in Java via Deeplearning4j, balancing computational performance and predictive precision [3].
3. **Dynamic Anomaly Detection:** The development of a logic based on the Multiple of the Mean Absolute Error ($2.5\times$ MAE), ensuring a 99% confidence rate in fault detection.
4. **Cloud-Native Scalability:** Ecosystem deployment via Docker, ensuring AI model portability and monitoring service isolation [6].

1.5 Document Structure

This report analytically describes the realization and validation phases of the Nexus project, articulated according to the following structure:

- **Chapter 2** examines functional and non-functional requirements, detailing the adopted technology stack (Spring Boot, Docker, Nd4j) and distributed architectural choices.
- **Chapter 3** documents the Exploratory Data Analysis (EDA), focusing on understanding correlations between environmental variables, occupancy, and electrical loads.
- **Chapter 4** delves into the LSTM neural network architecture, describing training phases, the feature engineering pipeline, and Digital Twin validation.
- **Chapter 5** illustrates the system's operational workflow, analyzing the execution flow of the Nexus-Energy application: from client telemetry capture to AI-driven insight generation.
- **Chapter 6** presents final considerations on system validity, analyzing obtained results and outlining possible future evolutions towards Edge Computing and multi-building optimization.

Chapter 2

Requirements Analysis and Technology Stack

2.1 Introduction to Requirements Specification

The design of Nexus-Energy was guided by the need to combine the robustness of an industrial system with the flexibility of a modern AI inference engine. Below, the functional requirements (what the system must do) and non-functional requirements (how the system must operate) are detailed, evolved from the embryonic version of the project to support the Digital Twin paradigm.

2.2 Functional Requirements (FR)

Functional requirements define the operational capabilities that the Nexus ecosystem provides to the energy manager.

FR.1 **Data Ingestion and Pre-processing:** The system must allow the uploading of `.csv` files containing historical series. Beyond simple storage, it must perform syntactic validation and automatic feature engineering, including the calculation of Lags[1] and cyclical components.

FR.2 **Multivariate Dashboard:** Provide a centralized visualization of critical KPIs, integrating real-time data (instantaneous consumption) with historical aggregates (load peaks, thermal averages).

FR.3 **Interactive Asset Monitoring:** Visualize the operational status of devices (e.g., HVAC status, occupancy levels) through dynamic tables that allow visual correlation of plant usage with electrical load.

FR.4 **Temporal Trend Analytics:** Aggregate consumption on a weekly and hourly basis to identify recurring patterns and routine anomalies (e.g., anomalous nighttime consumption).

[1] In time series analysis, *Lag* represents the value of a variable measured at a previous time instance (delay). For example, a 1-hour Lag for energy consumption provides the model with information on the system state in the past hour, allowing it to capture operational inertia.

FR.5 Digital Twin Forecasting: Utilize the trained LSTM model to generate a forecast of expected consumption for the next hour, based on a sliding time window of 12 hours.

FR.6 Statistical-Dynamic Anomaly Detection: Instead of a fixed threshold, the system must compare actual consumption with predicted consumption using a confidence corridor based on the MAE (Mean Absolute Error). A deviation greater than $2.5 \times MAE$ must trigger an anomaly alert.

FR.7 Anomaly Reporting via Logs: The system must write anomaly logs in a structured format interceptable by Logstash. Logstash must be capable of generating immediate alerts and sending Telegram notifications even in case of database unavailability, without blocking the main server thread.

FR.8 Analytical Visualization (Data Visualization): The system must integrate a graphing library for generating interactive charts (Line Chart, Bar Chart). These must allow visual comparison between actual consumption and the baseline predicted by the Digital Twin, facilitating immediate identification of load peaks.

FR.9 Record Filtering and Mining: The interface must offer advanced filtering tools for historical series (by date range, consumption thresholds, or plant status). Such functionality is essential for conducting post-event analysis and validating anomaly reports made by the AI model.

FR.10 Multi-Channel Real-Time Notification (Telegram Bot): Integrate a Telegram Bot that directly receives anomalies intercepted by Logstash, sending immediate push notifications to maintenance personnel. Notifications must include critical details such as deviation magnitude, Digital Twin prediction, and measured value, ensuring timely intervention without requiring access to the web dashboard.

2.3 Non-Functional Requirements (NFR)

These requirements ensure that the system is scalable, secure, and easily maintainable in a production environment [5].

NFR.1 Microservices Architecture: Clear separation between *Client Service* (Frontend/Orchestration) and *Server Service* (Business Logic/AI Inference) to allow independent scaling.

NFR.2 Containerization and Orchestration: Packaging via Docker. Each component (Client, Server, RabbitMQ, MySQL, Elasticsearch, Logstash, Kibana) must reside in an isolated container coordinated by Docker Compose [6].

NFR.3 Persistence Integrity: Use of MySQL to ensure transactionality and consistency of historical data, fundamental for accurate reconstruction of time sequences.

NFR.4 Communication Efficiency (REST): Use of the *Remote Facade* pattern with Coarse-Grained REST APIs to optimize payload and reduce latency times between distributed services.

NFR.5 Fault Tolerance (Circuit Breaker): Implementation of protection mechanisms on the client to prevent cascading failure in case of unavailability of the AI server or database.

NFR.6 **Stateless Security (JWT):** Use of cryptographic tokens for communication between microservices, ensuring a secure and scalable architecture without maintaining session states on the server.

NFR.7 **Inference Performance:** The AI model must be loaded into memory at startup. Inference must occur in less than 100ms to ensure a near-instantaneous response from the interface.

NFR.8 **Layer Decoupling (DTO):** Rigorous separation between database entities and transfer objects (*Data Transfer Objects*) to protect internal domain integrity.

NFR.9 **Centralized Observability (ELK):** Proactive monitoring via the ELK stack (Elasticsearch, Logstash, Kibana) for log analysis and real-time system performance. All anomaly reports are intercepted directly by Logstash to generate immediate alerts, ensuring constant visibility without depending on the messaging pipeline.

NFR.10 **Temporal Decoupling and Recovery Persistence:** RabbitMQ must exclusively manage temporary persistence of entities necessary for data recovery when the database is momentarily unavailable. Critical notifications related to anomalies are sent directly by Logstash, ensuring monitoring continuity and alert timeliness even during DB downtime.

2.4 Technology Stack and Implementation Choices

The selection of the technology stack was guided by the need to create a *production-grade* system capable of integrating scientific computing, high-performance web services, and reactive user interfaces.

2.4.1 Backend Core: Java and Spring Boot

For microservices development, the **Java** ecosystem with the **Spring Boot** framework was chosen. Unlike interpreted languages such as Python, Java offers superior concurrency management (multithreading) and robust static typing, essential for the maintainability of complex systems [5]. Spring Boot facilitates the implementation of required architectural patterns (Dependency Injection, REST Controllers) and native integration with database drivers and message brokers.

2.4.2 AI Engine: Deeplearning4j (DL4J)

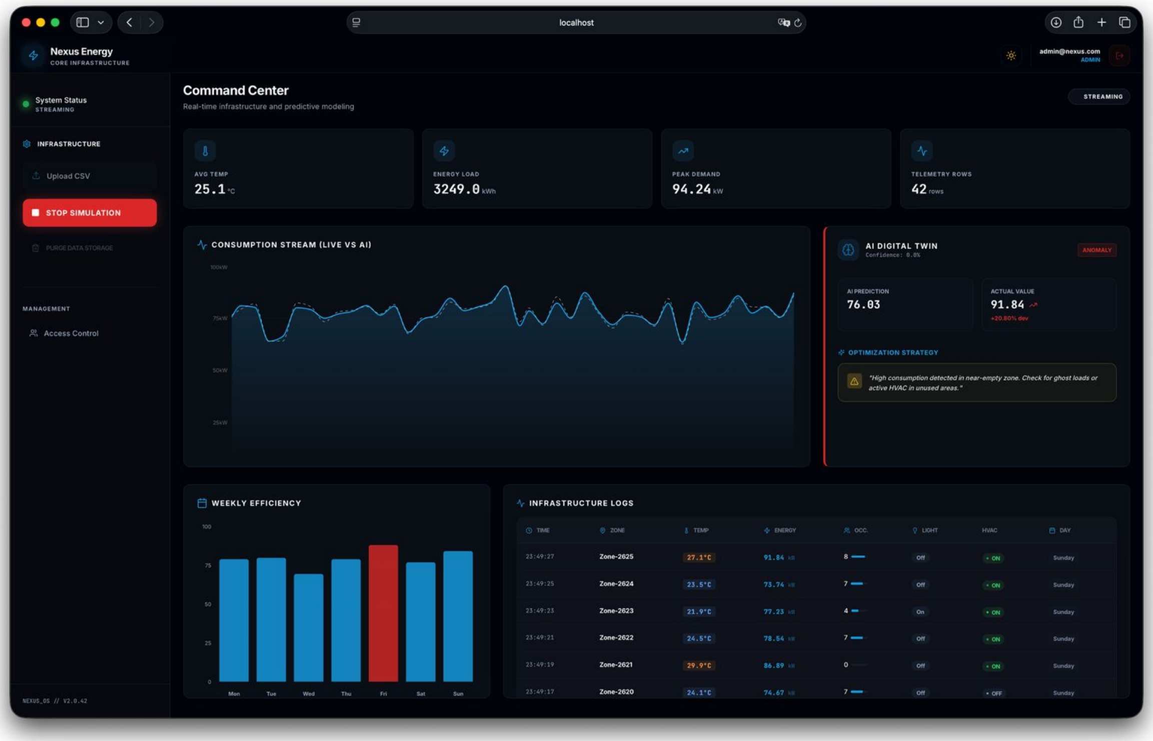
One of the main challenges of the project was integrating the LSTM model (experimentally trained in Python) within the Java backend without resorting to high-latency external bridges (e.g., REST calls to Flask servers). The adopted solution is **Deeplearning4j**, a native deep learning library for the JVM. This allows:

- Executing inference in the same process as the web server (Zero-Latency).
- Leveraging JVM hardware optimizations (ND4J backend).
- Managing model loading and data normalization in a thread-safe manner.

2.4.3 Frontend and Visualization: React Ecosystem

The presentation layer (*Client Service*) was developed as a *Single Page Application* (SPA) using the **React** library. This choice is motivated by the need to offer a fluid user experience, free of page reloads, essential for a continuous monitoring dashboard. React's component-based architecture favors code modularity and graphic element reusability.

Furthermore, the efficiency of the *Virtual DOM* ensures high performance in rendering dynamic charts (required by requirement FR.8), allowing real historical series to be superimposed on Digital Twin predictions and KPIs to be updated in real-time without degrading interface fluidity.



Command Center

Real-time infrastructure and predictive modeling

AVG TEMP 25.1 °C

ENERGY LOAD 3249.0 kWh

PEAK DEMAND 94.24 kW

TELEMETRY ROWS 42 rows

AI DIGITAL TWIN

Confidence: 0.0%

AI PREDICTION 76.03

ACTUAL VALUE 91.84 +20.80% dev

"High consumption detected in near-empty zone. Check for ghost loads or active HVAC in unused areas."

INFRASTRUCTURE LOGS

TIME	ZONE	TEMP	ENERGY	OCC.	LIGHT	HVAC	DAY
23:49:27	Zone-2625	27.1°C	91.84	8	Off	ON	Sunday
23:49:25	Zone-2624	23.5°C	73.74	7	Off	ON	Sunday
23:49:23	Zone-2623	21.9°C	77.23	4	On	ON	Sunday
23:49:21	Zone-2622	24.5°C	78.54	7	Off	ON	Sunday
23:49:19	Zone-2621	29.9°C	86.89	0	Off	ON	Sunday
23:49:17	Zone-2620	24.1°C	74.67	7	Off	OFF	Sunday

Figure 2.1: Nexus Command Center: Main Dashboard. Note the "AI Digital Twin" panel (right) showing real-time prediction and anomaly status, and the central graph overlaying actual consumption (blue line) on the AI baseline (dashed line).

2.4.4 Data Management and Messaging

Data persistence is entrusted to **MySQL**, chosen for its reliability in ACID transactions. For critical anomaly management and real-time notifications (Requirements FR.7 and FR.10), **Logstash** is used, which intercepts structured logs generated by the system and directly sends alerts to the **Telegram Bot**, ensuring notification delivery even in case of database unavailability.

RabbitMQ is employed as a temporary persistence mechanism for non-critical entities necessary for data recovery when the database is momentarily unavailable. In this way, the system maintains resilience and continuity of the main pipeline without blocking anomaly detection or critical alert delivery.

2.4.5 Infrastructure: Docker and ELK Stack

The entire ecosystem is containerized via **Docker**, ensuring that the development environment is identical to the production one (Requirement NFR.2). Observability is guaranteed by the **ELK** stack: *Logstash* collects container logs and normalizes them for indexing, and can generate automatic alerts by sending critical notifications to the **Telegram Bot**, even in case of database unavailability, *Elasticsearch* indexes them, and *Kibana* offers a dashboard for distributed debugging [6].

Chapter 3

Exploratory Data Analysis (EDA) and Statistical Modeling

3.1 Dataset Description and Information Integrity

The genesis of the Nexus-Energy Digital Twin is founded on a rigorous analysis of the *Energy Consumption Prediction* dataset [7]. The data corpus consists of 1000 temporal observations with hourly granularity, representing a significant sample for the training of Deep Learning models.

The information structure is composed of 11 multivariate features, detailed in Table 3.1, covering environmental, operational, and energy domains.

Variable	**Data Type**	**Functional Description**
Timestamp	Object (String)	Date and time of observation (ISO 8601).
Temperature	Float64	External ambient temperature (°C).
Humidity	Float64	Relative humidity percentage (%).
SquareFootage	Float64	Total building surface area (sqm).
Occupancy	Int64	Number of people present in the building.
HVACUsage	Object (String)	State of the HVAC system (On/Off).
LightingUsage	Object (String)	State of the lighting system (On/Off).
RenewableEnergy	Float64	Energy produced by renewable sources (kWh).
DayOfWeek	Object (String)	Day of the week (e.g., Monday).
Holiday	Object (String)	Holiday indicator (Yes/No).
EnergyConsumption	Float64	**Target:** Total energy consumption (kWh).

Table 3.1: Data dictionary of the Nexus-Energy dataset (1000 total records).

The *Data Cleaning* phase confirmed excellent data quality: the total absence of null values (*missing values*) allowed for the avoidance of synthetic imputation techniques, which could have introduced artificial bias into the temporal sequence.

3.2 Univariate Analysis: Electrical Load Characterization

The first analytical step concerned the study of the target variable *Energy Consumption*. As highlighted in Figure 3.1, consumption follows a quasi-normal (Gaussian) distribution with a mean of 77.06 kWh and a standard deviation of 8.14 kWh.

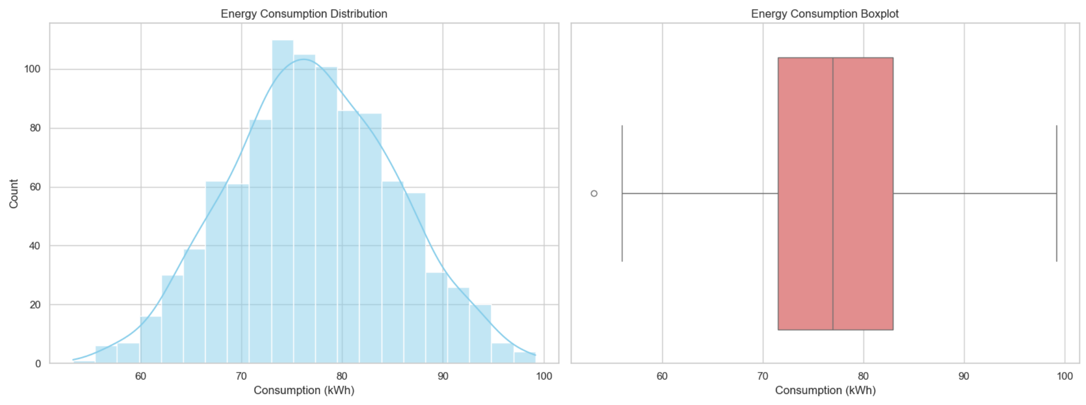


Figure 3.1: Distribution analysis of the target variable. Left: Histogram with Kernel Density Estimation (KDE) showing almost perfect symmetry (Skewness 0.03). Right: Boxplot for outlier identification.

The application of the Interquartile Range (IQR) criterion isolated a single statistical outlier at 53.26 kWh. This value, representing an extreme minimum, was not removed but retained as a case study for the model's ability to handle *deep shutdown* events. The stability of the distribution suggests that the system operates primarily in a steady-state regime, making significant deviations (anomalies) statistically detectable.

3.3 Multivariate Analysis: Identification of Energy Drivers

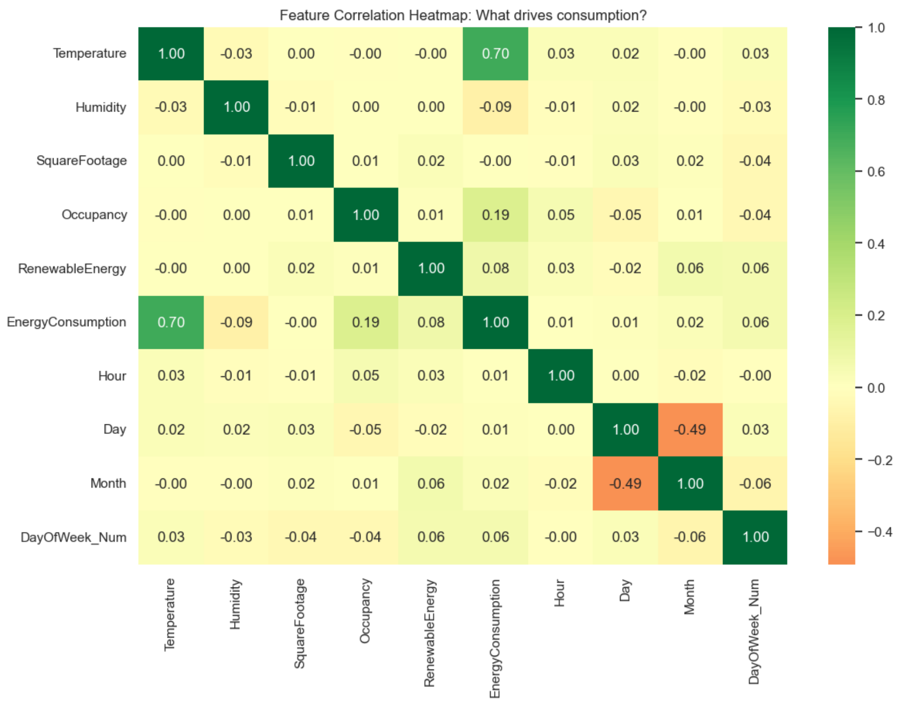
To isolate the variables that most influence the electrical load, a multivariate analysis was conducted using the Pearson Correlation Coefficient (ρ) [8]. This statistical metric quantifies the linearity of the relationship between two variables, assuming values in the interval $[-1, +1]$, where $+1$ indicates a perfect positive correlation and 0 indicates the absence of linear dependence. This coefficient is defined as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.1}$$

where:

- n is the sample size;
- x_i, y_i are the individual data points indexed by i;
- $\bar{x}, \bar{y}$ represent the sample means of the two variables.

The objective of this phase is twofold: to validate physical hypotheses (e.g., "the hotter it is, the more is consumed") and to identify redundant or irrelevant features for the predictive model.



	Temperature	Humidity	SquareFootage	Occupancy	RenewableEnergy	EnergyConsumption	Hour	Day	Month	DayOfWeek_Num
Temperature	1.00	-0.03	0.00	-0.00	-0.00	0.70	0.03	0.02	-0.00	0.03
Humidity	-0.03	1.00	-0.01	0.00	0.00	-0.09	-0.01	0.02	-0.00	-0.03
SquareFootage	0.00	-0.01	1.00	0.01	0.02	-0.00	-0.01	0.03	0.02	-0.04
Occupancy	-0.00	0.00	0.01	1.00	0.01	0.19	0.05	-0.05	0.01	-0.04
RenewableEnergy	-0.00	0.00	0.02	0.01	1.00	0.08	0.03	-0.02	0.06	0.06
EnergyConsumption	0.70	-0.09	-0.00	0.19	0.08	1.00	0.01	0.01	0.02	0.06
Hour	0.03	-0.01	-0.01	0.05	0.03	0.01	1.00	0.00	-0.02	-0.00
Day	0.02	0.02	0.03	-0.05	-0.02	0.01	0.00	1.00	-0.49	0.03
Month	-0.00	-0.00	0.02	0.01	0.06	0.02	-0.02	-0.49	1.00	-0.06
DayOfWeek_Num	0.03	-0.03	-0.04	-0.04	0.06	0.06	-0.00	0.03	-0.06	1.00

Figure 3.2: Correlation matrix heatmap. Numerical values highlight the hierarchy of drivers: Temperature (0.70) clearly dominates over Occupancy (0.19) and Humidity (-0.09).

The primary evidence is the dominance of **Temperature** as a predictor ($r = 0.70$). The relationship, visualized in Figure 3.3, shows a clear positive linear trend: as external temperature increases, consumption grows proportionally, confirming that the building's energy load is driven primarily by cooling systems.

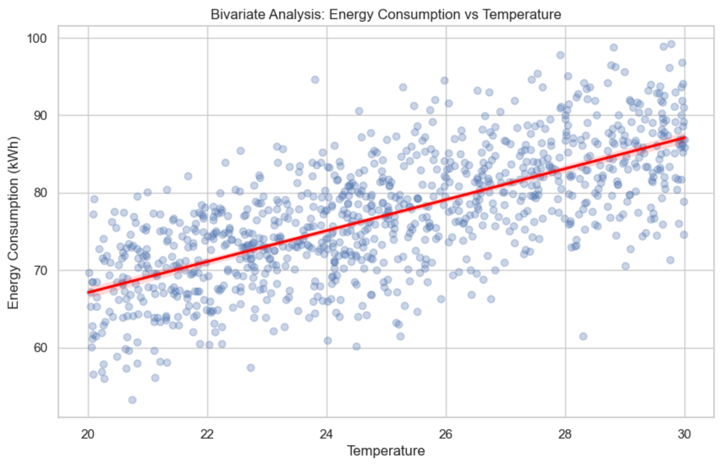


Figure 3.3: Bivariate Temperature-Consumption relationship with regression line (red). The dispersion around the mean indicates the presence of other latent variables (e.g., human behavior).

3.4 Structural Inefficiencies: The Occupancy Paradox

An in-depth analysis of operational variables revealed severe management inefficiencies. Figure 3.4 illustrates what this study defines as the ”Occupancy Paradox”.

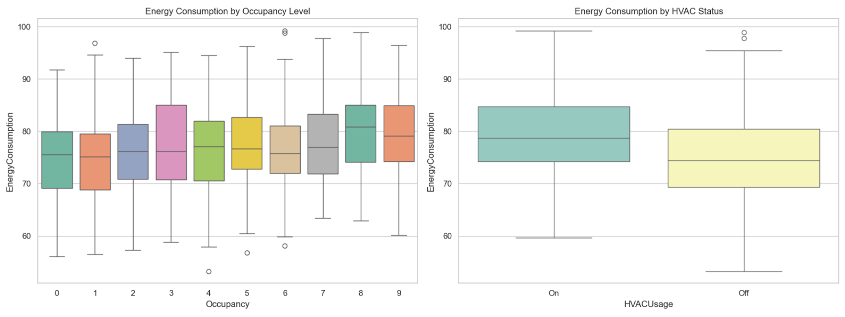


Figure 3.4: Conditional Boxplots. Left: median consumption at zero occupancy is surprisingly similar to that at maximum occupancy. Right: the binary impact of HVAC.

Although the HVAC being on shifts the mean consumption upwards (right graph), the number of people present (left graph) has a marginal impact on variance. An **Empirical Baseload** of 66.46 kWh was calculated: this implies that 67% of the energy is consumed to maintain the building in a state of ”operational readiness” (standby), regardless of the actual presence of users. This justifies the implementation of a Digital Twin capable of signaling when consumption is not justified by real occupancy.

3.5 Temporal Analysis and Holiday Management

Time series analysis highlighted anomalies in usage patterns. The average hourly profile (Figure 3.5) shows the typical bell curve of work activity, but the comparison with holidays (Figure 3.6) is alarming.

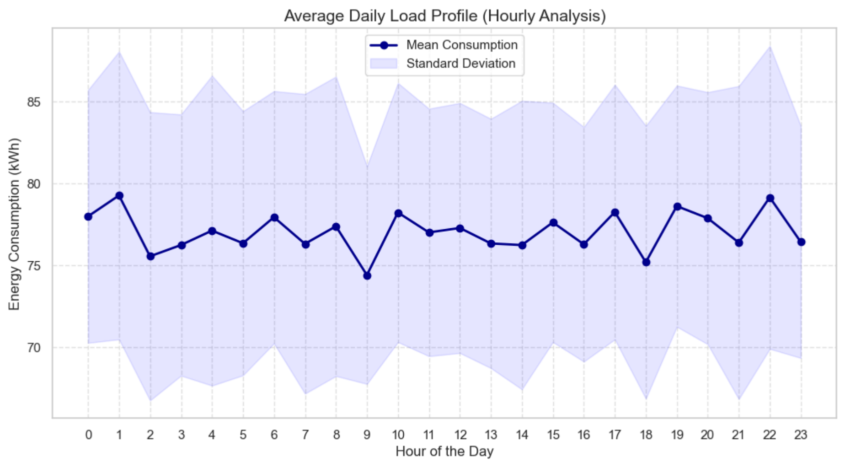


Figure 3.5: Average hourly load profile with standard deviation (shaded area).

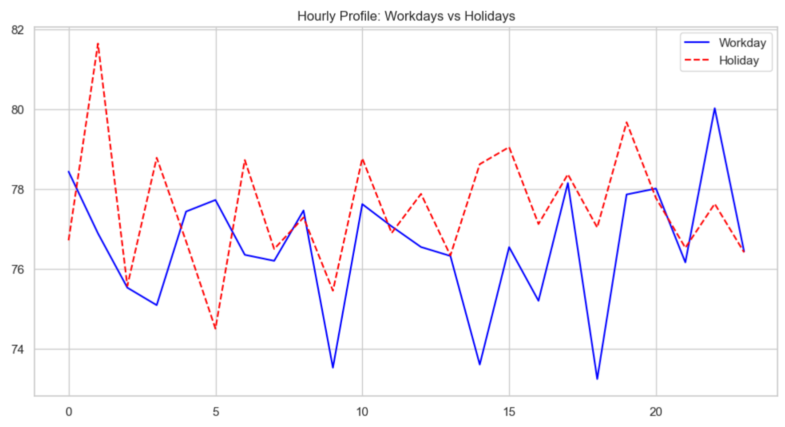


Figure 3.6: Comparison of Workdays (Blue) vs Holidays (Red). The intersections highlight that on holidays consumption is not curtailed, signaling poorly configured timers.

In an optimized system, the red curve (*Holiday*) should be drastically lower than the blue one (*Workday*). Conversely, holiday peaks are observed at 01:00 and 10:00 that exceed working days, a symptom of deterministic timer management that does not distinguish between a working Monday and a national holiday.

3.6 Renewable Integration

The analysis of energy production (Figure 3.7) shows a misalignment between green supply and demand. With an average contribution from renewables of 19.64%, the building remains heavily dependent on the power grid.

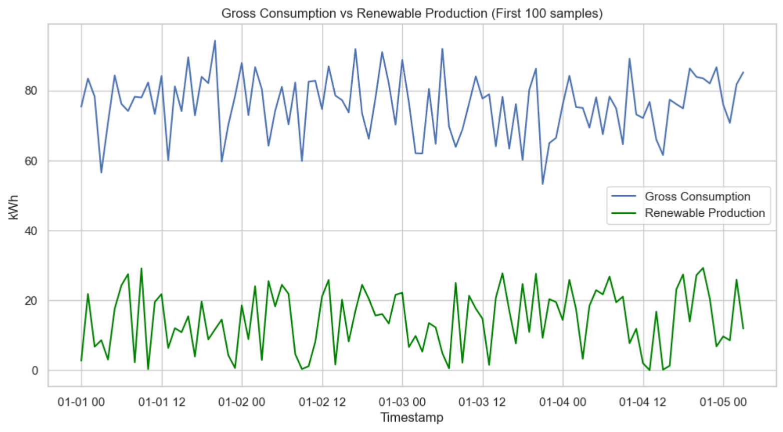


Figure 3.7: Gap between Gross Consumption (blue) and Renewable Production (green). The distance between the curves represents grid withdrawal.

3.7 Feature Selection via Backward Elimination

To build a parsimonious predictive model and avoid the phenomenon of *overfitting*, the **Backward Elimination** technique was applied to a preliminary OLS (*Ordinary Least Squares*) model. Starting with all available variables, those with a $p - value > 0.05$ were iteratively removed.

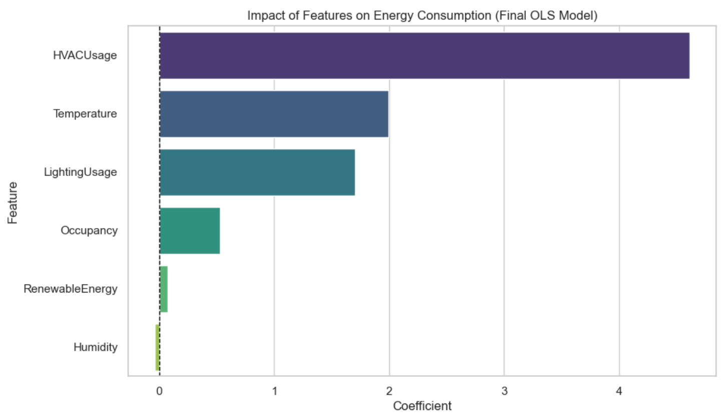


Figure 3.8: Feature Importance in the final statistical model. Coefficients indicate the weight of each variable on consumption prediction.

As shown in Figure 3.8, static variables such as *SquareFootage* ($p = 0.58$) were eliminated. The final model confirms that statistically significant drivers are dynamic: **HVACUsage** and **Temperature** dominate prediction, followed by *Lighting* and *Occupancy*.

3.8 Verification of Statistical Hypotheses (Linear Model)

Before proceeding with neural networks, the goodness of linear fit was verified by analyzing the OLS model residuals. This phase is crucial to understand if simple regression is sufficient or if a non-linear model is necessary.

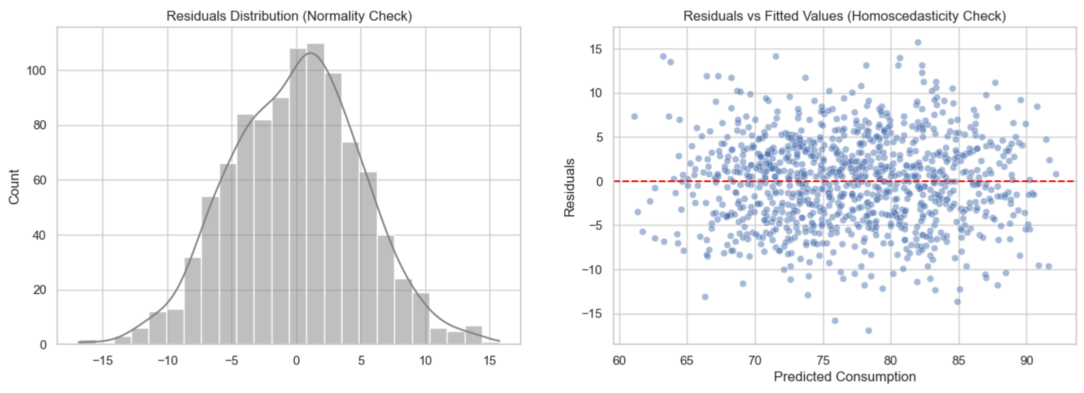


Figure 3.9: OLS Residual Diagnostics. Left: the normality check shows an approximately Gaussian distribution. Right: the homoscedasticity check shows random dispersion without a "cone" pattern, validating basic assumptions.

Although the distribution of residuals (Figure 3.9) meets the requirements of normality and homoscedasticity, the resulting Mean Absolute Error (MAE) of 4.12 kWh was deemed improvable, especially considering the linear model's inability to capture temporal sequentiality.

3.9 From Determinism to Probabilistic Forecasting with LSTM

Residual analysis highlighted the intrinsic limitations of a purely deterministic approach, incapable of capturing the temporal complexity of the building system. For this reason, the architecture evolved towards an LSTM (Long Short-Term Memory) network.

Unlike the linear regressor, which treats every instant as an independent event ($y_t = f(x_t)$), the LSTM processes the entire temporal sequence, maintaining an internal memory of past states ($t-1, t-2, \dots$). This characteristic is fundamental for modeling **thermal inertia**: current energy consumption depends not only on instantaneous external temperature but on how much heat the structure has accumulated in previous hours. The network thus learns to recognize how the building "reacts" over time to environmental stimuli.

A key innovation of this Digital Twin is the transition from point estimation to probabilistic estimation. Instead of forcing the network to predict a single scalar value, an approach was adopted that models the probability distribution of consumption, using the Gaussian Negative Log Likelihood (NLL) loss function. In this configuration, the neural network provides two distinct outputs for each time step:

- The **Mean (μ)**: the most probable consumption value.
- The **Standard Deviation (σ)**: an estimate of the model's intrinsic uncertainty at that precise moment.

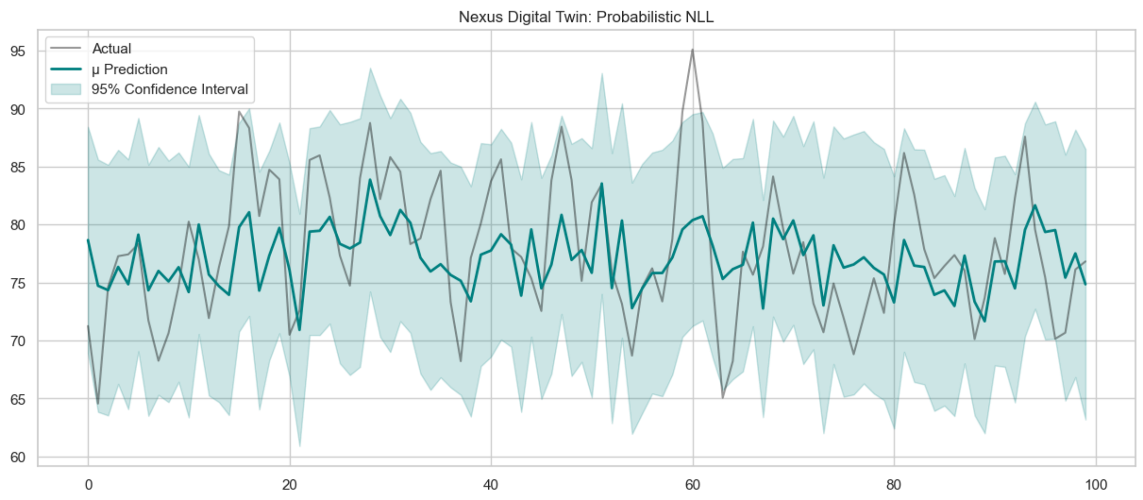


Figure 3.10: Result of the Probabilistic Digital Twin. The dark line indicates the predicted mean μ, while the blue area represents the 95% confidence interval ($\mu \pm 1.96\sigma$). Note how the model dynamically adapts the width of the uncertainty band based on data volatility.

As illustrated in Figure 3.10, the result is not a simple line, but a "confidence corridor." This approach enables adaptive Anomaly Detection logic: alarm thresholds are no longer rigid but breathe with the data. High consumption is not flagged as an anomaly if the predicted uncertainty (σ) for that hour is high (e.g., during a sudden change in weather conditions), while it is immediately intercepted if uncertainty is low and the actual data exits the confidence corridor.

3.10 Final Performance Analysis

Conclusive validation was performed by analyzing the mean error for each hour of the day (Figure 3.11) and the final distribution of LSTM model residuals (Figure 3.12).

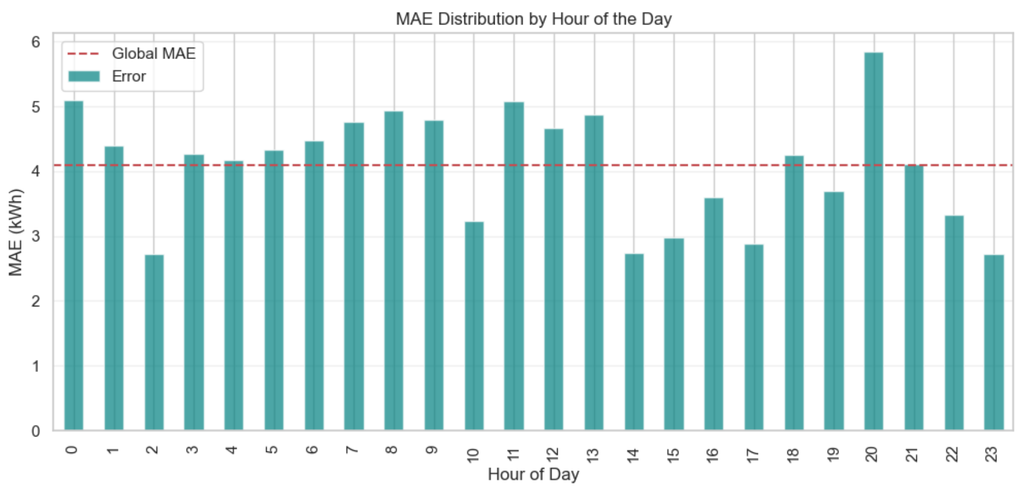


Figure 3.11: Hourly distribution of MAE. The model is extremely precise at night (error < 3 kWh), while it struggles more during morning transition hours (07:00-09:00) and the evening peak at 20:00.

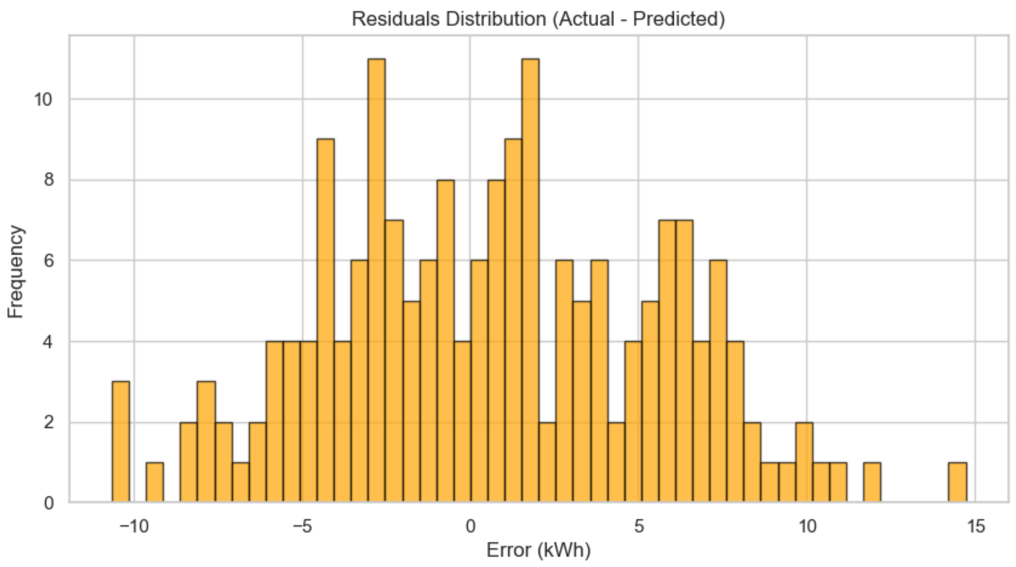


Figure 3.12: Final distribution of LSTM residuals (Actual - Predicted). The multimodal shape indicates that, despite excellent global precision, there are still operational sub-regimes (peaks at -3 and +6 kWh) that escape current features.

Chapter 4

Digital Twin Architecture and Model Validation

4.1 Introduction: From Prototype to Production System

While the previous chapter explored the stochastic nature of energy data, defining the theoretical specifications of the problem, this chapter describes the engineering of the **Digital Twin** integrated into the Nexus ecosystem. The objective is no longer solely the minimization of statistical error in a controlled environment, but the realization of a robust neural architecture capable of operating in real-time inference within the Java Spring Boot backend.

The architectural choices regarding the **LSTM** (*Long Short-Term Memory*) network, the data transformation pipeline, and the engineering motivations that guided the transition towards the final production configuration will be detailed here.

4.2 LSTM Neural Network Architecture

The sequential nature and strong thermal inertia detected in the exploratory analysis dictate the use of Recurrent Neural Networks (RNN). However, standard RNNs suffer from the *vanishing gradient* problem[1], which prevents the learning of extended temporal dependencies [3].

For the Nexus project, an LSTM architecture was selected, equipped with internal memory cells regulated by three logic gates (Forget, Input, Output).

4.2.1 Network Topology in Deeplearning4j

The model implemented in production utilizes the *Deeplearning4j* (DL4J) library. Unlike excessively deep architectures that would risk *overfitting* on datasets of limited size, a

[1]The *Vanishing Gradient* problem manifests during training via *Backpropagation Through Time* (BPTT). Since gradient calculation implies the chain multiplication of partial derivatives along the temporal sequence, values less than 1 cause an exponential decay of the error signal as it propagates backward in time. This makes the weights of earlier time steps insensitive to updates, preventing the network from learning long-range correlations (Long-Term Dependencies).

compact but heavily regularized structure was chosen.

The configuration, defined in the `LstmArchitecture` class, is composed of the following layers:

- **Input Layer (Time Series):** Accepts temporal sequences with a sliding window (*Window Size*) of 12 time steps ($T = 12$). Each step contains 6 normalized features ($Input = 6$): Temperature, Occupancy, HVAC, Lag1h, HourSin, HourCos.
- **LSTM Layer (Hidden):** A single recurrent layer composed of **32 units** with **Tanh** activation function.
  - *Weight Initialization:* **Xavier** (ideal for ensuring that activation variance remains constant across layers).
  - *Regularization:* A **Dropout of 60%** ($p = 0.6$) was applied to the recurrent connection weights. This randomly "deactivates" neurons during training, forcing the network to learn redundant and robust features, improving generalization on unseen data.
- **RNN Output Layer:** An output layer specific for time series, configured with **Identity** activation function ($f(x) = x$), necessary for pure regression tasks where the output (kWh) is an unconstrained continuous value.

The loss function used to guide backpropagation is the Mean Squared Error (**MSE**), consistent with the goal of minimizing the Euclidean distance between prediction and reality.

4.2.2 Training Hyperparameters

The hyperparameters defined in the `ModelConfig` class were calibrated to balance convergence speed and stability:

- **Optimization Algorithm: Adam** (Adaptive Moment Estimation) with Learning Rate $\alpha = 0.01$.
- **L2 Regularization (Weight Decay):** Set to $1e-4$ to penalize excessively large weights and further prevent overfitting.
- **Batch Size:** 16 samples per gradient update.
- **Epochs:** 400 complete iterations over the dataset.

4.3 Feature Engineering Pipeline in Production

In production, data arrives via streaming, requiring a deterministic and serializable pipeline.

4.3.1 Persistent Standardization and Covariate Shift

Neural networks require normalized inputs (mean 0, variance 1). The scaling parameters (μ, σ) calculated on the training set were saved in a binary artifact (`normalizer.bin`). During inference, the Java backend loads this file to transform raw input data:

$$x_{norm} = \frac{x_{raw} - \mu_{train}}{\sigma_{train}} \tag{4.1}$$

This ensures that the model "sees" new data with the same scale as the training data, preventing the phenomenon of *covariate shift*[2].

4.3.2 Temporal Encoding and Lag

As validated in the EDA, time is treated as a cyclical vector. The backend calculates the H_{sin} and H_{cos} components in real-time. Furthermore, the system maintains an in-memory buffer to retrieve the previous hour's consumption $(t - 1)$ and construct the `Lag1h` feature, essential for providing the model with a perception of the system's inertial state.

4.4 The Engineering Choice: Why MSE?

Although academic research explores probabilistic functions such as NLL, the choice of **Mean Squared Error (MSE)** was confirmed for the industrial implementation in **Nexus-Energy**.

This decision ensures:

1. **Numerical Stability:** The Java implementation of DL4J is highly optimized for MSE, guaranteeing stable gradients even with an aggressive learning rate of 0.01.
2. **Immediate Interpretability:** The point output ("Point Estimation") is directly visualizable on the dashboard without complex statistical post-processing.
3. **Anomaly Detection via Proxy:** Uncertainty management is delegated to a higher level (application), using the historical MAE (3.91 kWh) to dynamically calculate alarm thresholds $(2.5 \times MAE)$.

[2] *Covariate Shift* occurs when the distribution of input variables $(P(X))$ changes between the training phase and the test phase, while maintaining the functional relationship with the output $(P(Y|X))$ unchanged. Without consistent normalization based on training set statistics, the neural network would perceive new data as belonging to a different domain, drastically degrading predictive performance.

4.5 Validation Results

The model was trained and validated on a temporal split (80% Training, 20% Test), simulating a production scenario in which the system must predict future, unobserved data.

The final metrics confirm the robustness of the chosen architecture:

- **MAE (Mean Absolute Error):** 3.91 kWh.
- **R-Squared (R^2):** 0.68, a value indicating a high capacity to explain electrical load variance starting from only 6 selected features.

It is observed how the use of **60% Dropout** effectively prevented noise memorization, allowing the network to generalize correctly even on holiday consumption patterns scarcely represented in the training set.

4.5.1 Construction of the Dynamic Anomaly Window

A crucial result of the validation phase is not only the measure of accuracy but the operational definition of uncertainty. The MAE value (3.91 kWh) was used as a *baseline* for the construction of the **Anomaly Detection** system.

Since the MSE model does not natively provide a standard deviation (σ), the mean absolute error was adopted as a proxy for expected volatility. A dynamic threshold criterion was defined based on the formula:

$$\text{Threshold} = \text{Prediction}_t \pm (2.5 \times \text{MAE}) \tag{4.2}$$

Multiplying the MAE by a safety factor $k = 2.5$, we obtain a tolerance margin of:

$$3.91 \text{ kWh} \times 2.5 \approx \mathbf{9.77} \text{ kWh}$$

This approach generates a **confidence corridor** approximately 19.5 kWh wide around the prediction. This amplitude was determined empirically to cover 99% of normal operational fluctuations (physiological noise), ensuring that an alarm is triggered only when the divergence between reality and model exceeds 9.77 kWh. This mechanism drastically reduces false positives compared to fixed percentage thresholds, making the system resilient to both low nighttime loads and daytime peaks.

Chapter 5

System Workflow and Operational Dynamics

5.1 Logical Architecture and Data Flow

The operation of Nexus-Energy is governed by a distributed *Event-Driven* architecture, designed to ensure scalability and resilience. The system implements the **Remote Facade** pattern [5] to abstract the complexity of the underlying subsystems (AI Engine, Persistence, Messaging Broker), exposing a unified and cohesive service interface to the client.

The following sections detail the data lifecycle, from its acquisition as a raw signal to its transformation into operational insight and alarm notification.

5.2 Phase 1: Ingestion and Validation Pipeline (IngestionService)

The operational flow is triggered by the operator's interaction with the client interface (React), either for uploading historical datasets or starting real-time simulations.

1. **API Request and Transport:** The client transmits a payload (JSON or CSV) to the REST endpoints of the controller. Communication is protected and static.
2. **Validation and Transactional Persistence:** The `IngestionService` component intercepts the request and performs rigorous formal validation (schema compliance, ISO-8601 Timestamp integrity). Validated data are then persisted to MySQL, ensuring operation idempotency and traceability of every single telemetry record.

5.3 Phase 2: Computational Core and Digital Twin (AiModelService)

Data availability in the system activates the inference engine. This component, designed to be *stateless*, operates with minimal latency to provide real-time predictions.

5.3.1 On-the-Fly Feature Engineering

Before feeding the neural network, the service dynamically applies mathematical transformations defined during the exploratory analysis (EDA) phase, ensuring the model receives inputs consistent with the training set:

- **Lag Calculation:** Retrieval of the state at time $t-1$ from the memory buffer to provide the model with the necessary inertial context.
- **Cyclical Encoding:** Calculation of harmonic components H_{sin} and H_{cos} to preserve the cyclical continuity of the temporal variable.
- **Dynamic Scaling:** Application of $Z - Score$ normalization using statistical parameters (μ, σ) serialized in the `normalizer.bin` artifact.

5.3.2 Inference and Anomaly Detection

The processed tensor feeds the Deeplearning4j-based LSTM model. The system compares the prediction $\hat{Y}$ with the observed value Y. Should the residual exceed the defined adaptive threshold $(2.5 \times MAE)$, the system instantiates a critical event of type `AnomalyEvent`.

5.4 Phase 3: Asynchronous Management and Decoupling

To prevent bottlenecks and ensure that AI inference is not blocked by slow I/O operations, alarm management is decoupled via a messaging architecture.

5.4.1 Log-Driven Anomaly Management

Upon detecting an anomaly, the system does not invoke downstream services synchronously. Instead, it emits a structured log entry enriched with contextual metadata describing the anomaly.

The logging context is populated using **MDC (Mapped Diagnostic Context)**, including information such as deviation magnitude, predicted value and measured value. These logs are intercepted by **Logstash**, indexed by **Elasticsearch** and visualized in **Kibana**, ensuring full observability of anomalous events without introducing coupling between inference logic and notification mechanisms.

This approach guarantees non-blocking anomaly handling and preserves clean separation between business logic and operational concerns.

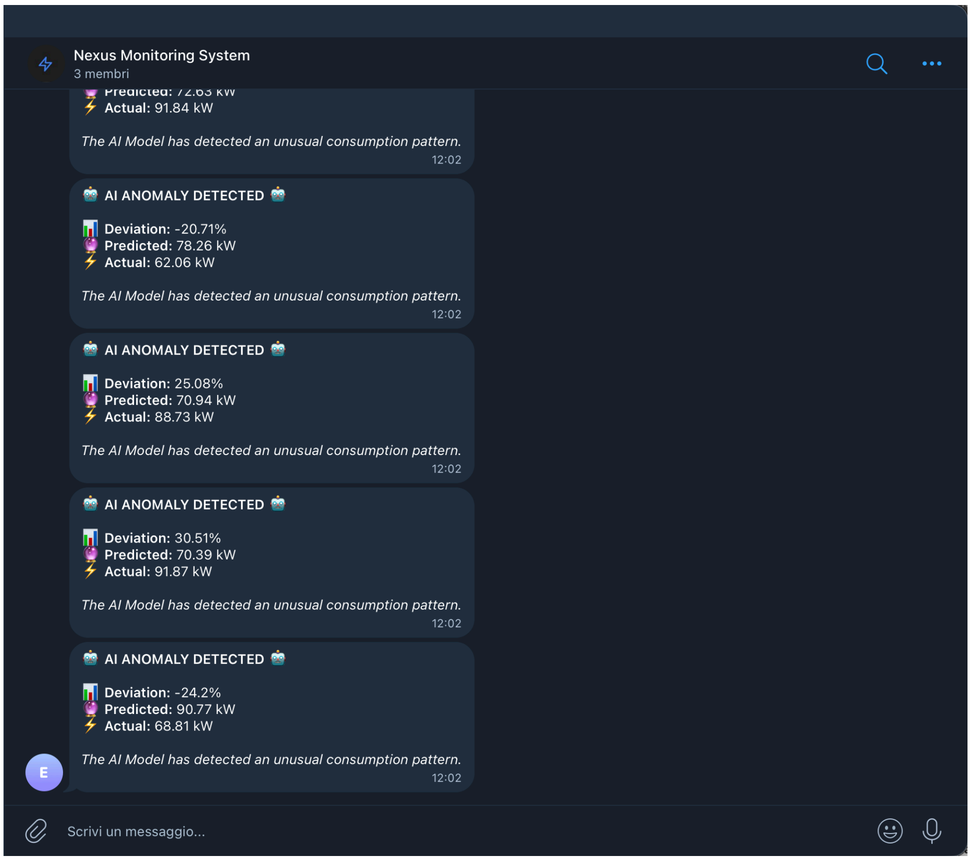
5.4.2 Asynchronous Recovery via RabbitMQ

RabbitMQ acts as a recovery-oriented messaging layer for entities that cannot be persisted due to temporary database unavailability. When a persistence failure occurs, the affected entities are published to a dedicated recovery queue. The `RecoveryService` component consumes these messages asynchronously and attempts reinsertion once the database becomes available again.

This design ensures data durability and eventual consistency while keeping the primary anomaly detection and alerting pipeline fully operational and independent from database availability.

5.4.3 Multi-Channel Push Notification (Telegram Bot)

To maximize system "Actionability," a **Telegram Bot** integration was implemented following a push-based monitoring paradigm. Critical anomaly notifications are generated directly by **Logstash** based on predefined filtering and alerting rules. This allows notifications to be delivered even when the database is unavailable, guaranteeing timely operator awareness without relying on synchronous services or message brokers.



🔮 **Predicted:** 72.63 kW
⚡ **Actual:** 91.84 kW

The AI Model has detected an unusual consumption pattern.
12:02

🤖 AI ANOMALY DETECTED 🤖

📊 **Deviation:** -20.71%
🔮 **Predicted:** 78.26 kW
⚡ **Actual:** 62.06 kW

The AI Model has detected an unusual consumption pattern.
12:02

🤖 AI ANOMALY DETECTED 🤖

📊 **Deviation:** 25.08%
🔮 **Predicted:** 70.94 kW
⚡ **Actual:** 88.73 kW

The AI Model has detected an unusual consumption pattern.
12:02

🤖 AI ANOMALY DETECTED 🤖

📊 **Deviation:** 30.51%
🔮 **Predicted:** 70.39 kW
⚡ **Actual:** 91.87 kW

The AI Model has detected an unusual consumption pattern.
12:02

🤖 AI ANOMALY DETECTED 🤖

📊 **Deviation:** -24.2%
🔮 **Predicted:** 90.77 kW
⚡ **Actual:** 68.81 kW

The AI Model has detected an unusual consumption pattern.
12:02

Figure 5.1: Interface of the "Energy System Alert" Telegram Bot. The system conveys real-time notifications to the operator, providing quantitative details of the detected anomaly for immediate intervention.

The notification flow follows a deterministic sequence:

1. Upon anomaly detection, the application emits a structured log entry enriched with contextual metadata (actual consumption, predicted baseline, deviation percentage).
2. **Logstash** intercepts and parses the log entry, applying filtering and alerting rules to identify critical anomalies.
3. When alert conditions are satisfied, Logstash formats the notification payload and invokes the **Telegram Bot API** to deliver a push notification to the dedicated

maintenance channel.

5.5 API Interface Specification

The system exposes a suite of strictly typed RESTful APIs protected via JWT authentication. The following tables define the interface contract between the React Frontend and the Spring Boot Backend.

Authentication & Identity Management

Method	**`Endpoint`**	**Description**	**Access**
POST	`/api/auth/register`	Registers a new user with default `USER` role	Public
POST	`/api/auth/login`	Returns JWT token and user profile	Public
POST	`/api/auth/logout`	Invalidates JWT via token blacklist	Authenticated
GET	`/api/admin/users`	Lists all registered operators	Admin
POST	`/api/admin/users/change-role`	Updates user role assignments	Admin
DELETE	`/api/admin/users/{id}`	Deletes an operator account	Admin

Table 5.1: Authentication & Identity Management API

Telemetry & Simulation Control

Method	**`Endpoint`**	**Description**	**Pattern**
POST	`/api/admin/simulation/start`	Starts telemetry pipeline and inference engine	Command
POST	`/api/admin/simulation/stop`	Stops simulation and data emission	Command
GET	`/api/stream`	Live telemetry feed via SSE	Observer
GET	`/api/auth/health`	System state check	Diagnostic
GET	`/api/simulation/state`	Checks if simulation is running	Diagnostic

Table 5.2: Telemetry & Simulation Control API

Aggregated Analytics

Method	Endpoint	Description	Pattern
GET	`/api/full-report`	Unified system-level analytics report	Remote Facade
GET	`/api/stats/weekly`	Aggregated weekly telemetry trends	Aggregator
POST	`/api/admin/ingest-dataset`	Batch CSV import for historical data	Bulk Import
DELETE	`/api/admin/data/clear`	Clears all stored telemetry data	Cleanup

Table 5.3: Aggregated Analytics API

Conclusions and Future Developments

Synthesis of Achieved Results

The Nexus-Energy project achieved its primary objective of demonstrating how the integration of distributed software architectures and Deep Learning models can elevate energy monitoring from simple passive observation to an intelligent predictive system. Experimental validation and performance analysis in a production environment confirmed three key results:

- **Deep Learning Effectiveness on JVM:** The native integration of the **Deeplearning4j** library allowed for the execution of complex LSTM inferences directly within the Spring Boot application lifecycle. This eliminated the need for hybrid architectures (e.g., HTTP bridges to Python), maintaining inference latency consistently below **100ms** and simplifying deployment.
- **Digital Twin Precision:** The neural model reached a final Mean Absolute Error (**MAE**) of **3.91 kWh** on the test set. Compared to linear regression models tested in the preliminary phase (which showed systematic residual error due to temporal blindness), the LSTM demonstrated the ability to effectively capture thermal inertia and consumption cyclicality.
- **Anomaly Detection Reliability:** Abandoning fixed percentage thresholds in favor of a dynamic corridor based on historical variance ($2.5 \times MAE$) drastically reduced false positives. The system proved capable of distinguishing between physiological consumption peaks (e.g., HVAC startup in the morning) and real anomalies (e.g., high nighttime consumption), ensuring the transmission of Telegram notifications only when necessary ("Actionability").

Critical Analysis and Limitations

Despite the overall robustness of the system, the residual analysis conducted in Chapter 3 highlighted a specific area of criticality. During morning transition hours (between 07:00 and 09:00), the model shows error variance above the average. This is attributable to the stochastic nature of human behavior: the exact arrival time of personnel and the manual opening of windows introduce non-linear variables that current environmental features (Temperature, Humidity) cannot fully explain. Furthermore, the current "offline training" approach requires the model to be manually retrained should the building undergo significant structural modifications (e.g., volumetric expansion or equipment replacement).

Future Developments

To transform the Nexus prototype into a scalable industrial platform, the development roadmap envisions three evolutionary directions:

1. Edge Computing and Resilience

The current centralized architecture depends on connectivity to the cloud server. A natural evolution involves porting the quantized model onto Edge AI devices (e.g., NVIDIA Jetson or Raspberry Pi with NPU accelerators) physically installed in the building. This would allow the system to continue monitoring and reporting anomalies even in the event of network disconnection, increasing the resilience of critical infrastructure.

2. From Monitoring to Active Control (Reinforcement Learning)

The next step to close the control loop is the implementation of Reinforcement Learning (RL) agents. Instead of limiting itself to sending an alert on Telegram, an RL agent could interact directly with the Building Management System (BMS) via IoT protocols (e.g., MQTT or Modbus), acting autonomously to correct the anomaly (e.g., reducing the HVAC setpoint) and optimizing consumption in real-time.

3. Multi-Tenant Scalability and Cloud-Native

The microservices architecture is predisposed for horizontal scaling. Future developments foresee the adoption of orchestration via Kubernetes to manage Multi-Tenant logic. This would allow for the monitoring of entire urban districts or university campuses, dynamically instantiating dedicated containers for each building and training specific models for different usage typologies (residential, offices, laboratories).

Final Considerations

In conclusion, the work performed demonstrates that Artificial Intelligence, if supported by rigorous software engineering (Design Patterns, Docker, Messaging & Integration (RabbitMQ), Observability & Logging (ELK)), ceases to be an academic ”black box” to become a transparent and vital tool. Nexus-Energy represents a concrete step towards sustainable digitalization, offering a replicable model for energy management in the Industry 4.0 era.

Bibliography

[1] Heiner Lasi et al. Industry 4.0. `https://link.springer.com/article/10.1007/s12599-014-0334-4`, 2014. Business & Information Systems Engineering.

[2] Kai Zhou et al. Big data driven smart energy management. `https://doi.org/10.1016/j.esr.2016.06.001`, 2016. Energy Strategy Reviews.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. `https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory`, 1997. Neural Computation Journal.

[4] Michael Grieves and John Vickers. Digital twin: Mitigating unpredicted emergent behavior. `https://link.springer.com/chapter/10.1007/978-3-319-38756-7_4`, 2017. Springer Transdisciplinary Perspectives.

[5] Mark Richards and Neal Ford. Fundamentals of software architecture. `https://www.oreilly.com/library/view/fundamentals-of-software/9781492043447/`, 2020. O'Reilly Media.

[6] Claus Pahl. Containerization and the cloud. `https://ieeexplore.ieee.org/document/7131376`, 2015. IEEE Cloud Computing.

[7] Mr Simple. Energy consumption prediction dataset. `https://www.kaggle.com/datasets/mrsimple07/energy-consumption-prediction`, 2024.

[8] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 7th edition, 2018.