POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

PREDator: An Open-Access Deep Learning Framework for Long-Term Industrial Time Series Forecasting

Tesi di Laurea Magistrale in
Chemical Engineering - Ingegneria Chimica

Antonio Mazzitelli, 10801280

Advisor:
Prof. Flavio Manenti

Co-advisors:
Luis Felipe Sánchez

Academic year:
2025-2026

Abstract: The transition toward Industry 4.0 has established data-driven forecasting as a cornerstone for enhancing operational reliability and optimizing predictive maintenance strategies. A preliminary study in this field, exploiting Gaussian Process Regression (GPR), has demonstrated the feasibility of long-term predictions, with increasing accuracy when decreasing forecast's horizon. To further enhance forecasting performance, this work proposes a Deep Learning framework based on Long Short-Term Memory (LSTM) networks. Although LSTMs, as artificial neural networks, require substantial datasets to achieve high generalization, such data volumes are increasingly available within modern industrial infrastructures. Thanks to the employment a direct multi-step forecasting strategy, the proposed LSTM architecture effectively captures long-range dependencies while avoiding the error accumulation typical of recursive models. The results obtained in this context proved superior to GPR baselines, as the LSTM's ability to model complex non-linear temporal patterns provides greater stability and precision over extended horizons. The proposed model reaches a substantially lower RMSE, with reductions of up to an order of magnitude. The methodology was validated through two distinct industrial case studies: a batch fermentation process and an industrial furnace subject to fouling. This dual validation confirms the robustness and adaptability of the framework across different sectors. Finally, the validated methodology has been integrated into PREDator, an open-access Python-based application with a graphical user interface designed to streamline the deployment of advanced predictive tools in both industrial and academic environments.

Key-words: Predictive Models, Machine Learning, Predictive Maintenance

1. Introduction

Predictive modelling has become a cornerstone of modern industrial operations, enabling the conversion of large volumes of historical and real-time data into actionable forecasts. As highlighted by Lee et al., predictive capabilities are a fundamental element of cyber-physical industrial systems and play a crucial role in improving operational reliability and strategic planning [31]. *Heat exchanger fouling is a representative case where process-variable forecasting can significantly improve efficiency.* Indeed, fouling has a substantial economic impact: a survey by GE (General Electric) indicated that a single process unit with 5–10 exchangers subject to fouling could incur costs ranging from $250,000 to $2 million per year due to efficiency loss; to mitigate these costs,

sophisticated detection tools have been developed (Prasad et al., 2009 [46]). The industrial relevance of this domain is evidenced by the dominance of major software vendors in the Asset Performance Management (APM) market. Currently, GE Vernova offers proprietary APM suites that utilize "SmartSignal" technology and Digital Twins to detect anomalies and reduce unplanned downtime [54]. Similarly, Oracle provides the Oracle Predictive Maintenance solution within its Fusion Cloud ecosystem, which integrates "IoT Intelligent Applications" to monitor asset health and predict failures via proprietary machine learning algorithms [40]. However, while these commercial platforms offer robust capabilities for industrial deployment, they often function as "black boxes" with prohibitive licensing costs and closed architectures Their proprietary nature restricts access to underlying algorithms and data structures, limiting their utility for open scientific research and reproducibility. Moreover, student licences are not available for such tools, since they were conceived as pure SaaS to support the process operations management, rather than frameworks for research.

Typically, predictive models contemplate two different time horizons, each associated with distinct objectives and operational requirements. **Short-term** forecasts capture rapid process fluctuations and are crucial for predictive control, enabling quick corrective actions that maintain stability and optimize performance in real time. **Long-term** forecasts, on the other hand, reveal slow-evolving trends such as equipment degradation, making them indispensable for applications like predictive maintenance. In this regard, a comprehensive review by Jardine et al. (2006) demonstrates how predictive approaches fundamentally enhance decision-making in maintenance and asset management by allowing practitioners to anticipate future states of complex systems rather than react to failures after they occur [29]. Nevertheless, the literature addressing robust and reliable *long-term predictive models* remains relatively limited, particularly in complex industrial settings.

Mechanistic models can offer deep insight into process behavior, but developing them for complex industrial systems is often prohibitively difficult. Chemical plants involve nonlinear interactions, multi-phase phenomena and equipment-specific characteristics that are challenging to represent accurately through fundamental equations alone [20]. For example, even the fouling of a seemingly simple shell-and-tube heat exchanger can be extremely difficult to model from first principles. The deposition of carbon or fouling on tube surfaces gradually reduces heat transfer coefficients, while the turbulence of the fluid flow influences both heat transfer and fouling distribution. Typically such problems can be solved with accurate CFD simulations, but this usually requires excessive expenditures [27]. Moreover, detailed physical models typically require extensive calibration and may still fail to capture real-world variability [13]. Data-driven methods, by contrast, leverage the large volumes of historical and real-time operational data already available, allowing predictive models to learn system behavior directly from observations, making them more practical, scalable, and adaptable to process environments [7].

In the current literature, many studies report predictive models achieving very high accuracy. However, these results often apply primarily to one-step-ahead predictions. For instance, Pan et al. (2023) present a model for forecasting CO_2 concentration and flue gas flowrate that demonstrates excellent performance [41]. From the reported figures, the predicted values almost perfectly overlap with the true observations, and the error remains nearly constant and close to zero across the forecasting horizon. A similar pattern is observed in the work of Luo et al. (2025), who propose a transformer-based architecture for predicting light olefin yields in the MTO process and report outstanding predictive accuracy under a one-step-ahead forecasting setting [34]. This tendency is not confined to chemical-process applications, as several recent contributions in the broader time-series forecasting literature report comparable results [6, 11, 21]. These studies all rely on a rolling one-step-ahead evaluation strategy, in which the model is trained and tested to predict only the next time step, i.e., $\hat{x}_{t+1}$ from x_t, and the true observed value x_{t+1} is then used as input to compute the subsequent prediction $\hat{x}_{t+2}$. While this evaluation strategy often yields extremely high accuracy, it introduces a form of data leakage[5], since each forecast exploits information from the test set that would not be available in a real deployment scenario. Consequently, although such models are entirely appropriate for short-term or explicitly one-step-ahead prediction tasks —often explicitly the authors' intended scope— they are inherently unsuitable for long-term forecasting, where predictive errors should naturally accumulate instead of being artificially corrected by injecting future observations.

Numerous models have been developed for long-term time series prediction. For example, probabilistic models such as Gaussian Process Regression (GPR) have recently gained attention due to their flexibility in handling complex data series and their inherent ability to prevent overfitting [17]. In this context, Galeazzi et al. (2024) [19] applied GPR to predictive maintenance in an industrial furnace within the thermal deasphalting section of a used-oil regeneration plant. Another widely used and historically significant approach is the Auto-Regressive Integrated Moving Average (ARIMA) model [9, 56]. Its popularity stems from its simplicity and interpretability, which has made it a common choice in applications such as stock price prediction [2, 57]. However, traditional statistical models like ARIMA rely on linear assumptions and often struggle to capture complex temporal dependencies. The growth in available data and computational power has fueled the adoption

of deep learning techniques, particularly recurrent neural networks (RNNs), for modeling nonlinear time series dynamics [51]. For instance, Hossain et al. (2025) show that RNNs, specifically LSTMs, outperform classical ARIMA models in long-term renewable energy forecasting by effectively capturing intricate nonlinear temporal patterns [24]. Despite these advancements in other fields, there is a notable lack of applications utilizing advanced RNN models for long-term forecasting of time series of data collected from chemical plants.

This work aims to address this gap by proposing a generalized and automated framework for accurate long-term forecasting of complex industrial time series. Two case studies are considered: a batch fermentation process and the industrial furnace of the thermal deasphalting section. Moreover, one of the research objectives of this work is the definition of a Python-based framework, named `PREDator`, equipped with a graphical user interface (GUI), with the purpose of enabling users to load a dataset and execute the proposed model without requiring programming expertise. The inclusion of a compiled version is considered as a design objective, particularly relevant in industrial and academic contexts, as it aims to simplify deployment and mitigate potential setup-related issues. Overall, the framework is intended to provide access to the proposed methodology while abstracting its underlying complexity.

2. Case studies description

In this work two different case studies are considered to train and test several predictive models: a batch fermentation process and the industrial furnace of the thermal deasphalting section.

2.1. Batch fermentation

The dataset considered in this study refers to to the *Saccharomyces cerevisiae*, commonly known as baker's yeast, fermentation process. The system is operated in fed-batch mode, which is characterized by non-linear dynamics and continuous accumulation of mass. Regarding the physical layout, the operation is conducted in an aerated stirred tank reactor. Because yeast growth is an exothermic biological process, the reactor requires an active cooling mechanism, such as a thermal jacket or internal heat exchange coils, to dissipate excess heat and strictly regulate the temperature. Furthermore, the fed-batch strategy dictates that the primary carbon source is not supplied all at once; rather, the molasses feed is continuously pumped into the vessel over the course of the operation to prevent metabolic bottlenecks. As a direct consequence of this continuous liquid inflow with no corresponding outflow, the tank level rises monotonically from its initial starting volume until the batch is harvested.
While the complete dataset comprises a wider array of chemical and physical parameters, our methodology intentionally isolates a restricted subset of it. Accordingly, we considered only three key variables for our study: the molasses feed rate (*Molasses Feed*), the total volume of the system (*Tank Level*), and the temperature (*Temp*). The carbon source addition dictates the biological growth rate, the tank level reflects the integral mass balance of the fed-batch operation, and the temperature represents a critical environmental condition for the yeast culture.
The complete dataset is released at `https://github.com/industrial-data/predictor-explainer/tree/1b9ef07124e6c458957d6f7e122db96e61e4b346/data`.

2.2. Industrial furnace in an oil regeneration plant

The second case study considered is the industrial furnace PH-401B, a critical unit operating within the thermal de-asphalting section of the Itelyum Regeneration S.p.A. plant located in Pieve Fissiraga (LO), Italy. This facility processes approximately 200 kt per year of used mineral oils to produce regenerated lubricant bases through the proprietary Revivoil technology. The furnace plays a pivotal role in this process by heating the oil to a target temperature of $360°C$ to facilitate subsequent vacuum distillation. This unit has already sparked the interest of Galeazzi et al. (2024) [19], who addressed the problem of predicting the evolution of its operating point for predictive maintenance to optimize scheduling and reduce economic losses associated with downtime.
The furnace performance is governed by the progressive accumulation of **coking** (fouling) on the internal tube surfaces, an aging phenomenon that increases flow resistance and affects thermal efficiency. This degradation necessitates a scheduled maintenance shutdown, or a specific "blowing" procedure, a manual intervention to clear obstructions, when the internal pressure drop reaches a critical operational threshold of approximately 10 barg. Currently, the decision to halt operations for cleaning is managed by the operations team based on reaching this limit, rather than through long-term time-optimization strategies.

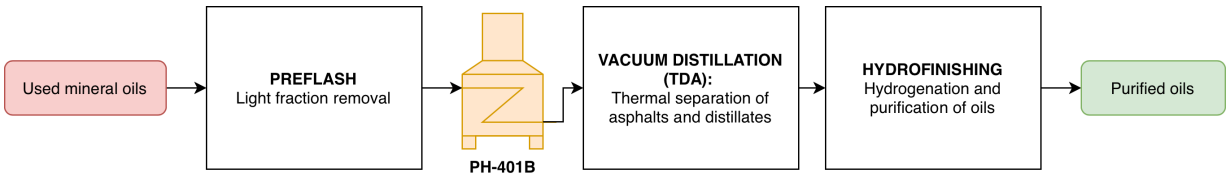


Figure 1: Block Flow Diagram of the thermal de-asphalting section at the Itelyum Regeneration S.p.A. facility in Pieve Fissiraga. The diagram shows the sequence of process steps—preflash, heating in furnace PH-401B to 360 °C, and subsequent vacuum distillation, used to treat approximately 200 kt/year of used mineral oils for the production of regenerated lubricant bases.

Historical time-series data was used, extracted from the Yokogawa Exaquantum Plant Information Management System (PIMS) to develop a predictive model. The key performance indicators monitored were variables directly affected by the coking process: Methane Fuel Consumption (FI-4091), which reflects the energy required to maintain the process temperature; Tube Skin Temperature (SK-4072), used to monitor local thermal stresses; and the critical prognostic variable, the Inlet Pressure Drop (PI-4301). The dataset consists of many variables recorded for the whole 67-days run (where the final pressure reached 11 barg), even though only the three mentioned above were considered to maintain model focus and reliability in forecasting the unit's remaining useful life.

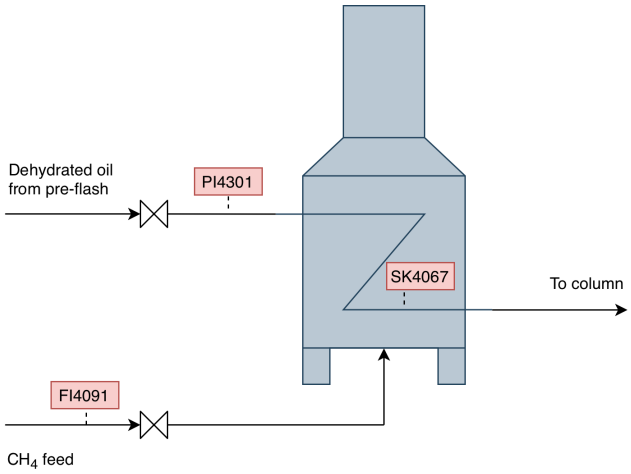


Figure 2: Simplified diagram of furnace PH-401B, showing the dehydrated oil feed, methane fuel supply, and monitored instrumentation relevant to coking-induced degradation. The key variables used in the study, fuel flow (FI-4091), inlet pressure drop (PI-4301), and tube skin temperature (SK-4067), reflect the progressive fouling of the coils, which increases flow resistance and reduces thermal efficiency. [19].

3. Methods

This section delineates the methodological framework and the computational pipeline developed for industrial time-series forecasting. The approach is structured as a progressive workflow: it begins with the refinement of raw data, moves through the establishment of statistical benchmarks, and culminates in the implementation of an advanced deep learning architecture. The methodology is divided into three primary stages. First, the Data Processing phase describes the signal cleaning, normalization, and augmentation techniques used to prepare the datasets. Second, we introduce the Baseline Models, specifically Polynomial Regression (PR) and Gaussian Process Regression (GPR), which serve as benchmarks for evaluating the benefits of non-linear modeling. Finally, we detail the Long Short-Term Memory (LSTM) network architecture, highlighting its design for direct multi-step forecasting and its capacity to manage long-term temporal dependencies.

3.1. Data preprocessing

The preprocessing pipeline begins with the application of a Simple Moving Average (SMA) to smooth short-term fluctuations in the raw signal[30]. It was demonstrated that this simple yet efficient technique can be an optimal

choice for similar tasks[52]. For a given variable x_t, a moving average with window size L is computed as

$$\mathrm{SMA}_t = \frac{1}{L} \sum_{i=0}^{L-1} x_{t-i}.$$

Outliers are then detected by comparing each observation to its local moving-average baseline. A value is flagged as anomalous when its absolute deviation exceeds a threshold defined as two standard deviations of the original signal:

$$|x_t - \mathrm{SMA}_t| > 2\sigma_x.$$

The 2 σ strategy was adopted after an empirical evaluation of the considered datasets, being strict enough to remove outliers while preserving noise. Detected outliers are replaced with missing values and subsequently interpolated to preserve temporal continuity and avoid artificial discontinuities in the learning process. This process is usually referred as *Missing Value Imputation* (MVI), and SMA was demonstrated to be a good choice for the interpolation of missing values[39]. After cleaning the series, an exponential moving average is applied to reduce noise[43]. The exponentially weighted moving average (EWMA) is defined as:

$$\mathrm{EWMA}_t = \alpha x_t + (1-\alpha)\,\mathrm{EWMA}_{t-1}, \qquad \alpha = \frac{2}{L+1}.$$

Finally, the filtered time series is normalized using Min–Max scaling to map all values into the interval $[0, 1]$. For each data point x_t, the transformation is given by

$$x_t^{\mathrm{scaled}} = \frac{x_t - x_{\min}}{x_{\max} - x_{\min}},$$

where $x_{\min}$ and $x_{\max}$ represent the minimum and maximum of the filtered series, respectively. This normalization step stabilizes the training of the neural network and ensures that all inputs contribute proportionally to the learning process, and it was proven to be effective for such problems[45].

3.2. Forecast horizon definition

The forecast horizon, denoted as H, refers to the specific sequence of future time steps that a model is required to predict based on historical observations[32]. It represents a fundamental dimension of the forecasting problem, directly influencing both the computational complexity of the learning task and the practical utility of the predictions.

While short-term forecasting is typically well-bounded, aiming to predict immediate future values, the definition of "long-term" forecasting remains ambiguous and highly domain-dependent in the literature [44]. In recent Long-Term Time Series Forecasting (LTSF) benchmarks, horizons are often fixed arbitrarily (e.g., 96, 192, 336, or 720 time steps) regardless of the data frequency [59]. However, a fixed numerical benchmark implies vastly different predictive difficulties depending on the sampling rate (e.g., a 96-step horizon represents 4 days for hourly data but 8 years for monthly data). This inconsistency makes it difficult to evaluate model robustness across heterogeneous datasets.

To address this ambiguity and ensure a consistent evaluation difficulty, we adopt a *seasonality-relative definition*. Following established forecasting principles that link evaluation horizons to the structural properties of the series [26], we define the look-ahead period relative to the intrinsic seasonality of the data. This ensures that the predictive challenge remains consistent regardless of the sampling frequency.

Let τ represent the primary seasonal cycle length of a given dataset (e.g., $\tau = 24$ for hourly data with daily periodicity). We evaluate our models across two distinct temporal scales:

- **Short-term Horizon (H_{short}):** Defined as 25% of the seasonal cycle ($H = 0.25\tau$). This horizon targets immediate fluctuations and local trends within a single cycle.
- **Long-term Horizon (H_{long}):** Defined as 50% of the seasonal cycle ($H = 0.5\tau$). This represents a more complex forecasting task, requiring the model to capture mid-range dependencies and the structural progression of the series without relying solely on immediate proximity to the input window.

3.3. Polynomial Regression

Polynomial regression models were employed to characterize the deterministic trend of the observed sensor data. This approach treats the time-series forecasting problem as a regression task where the target variable $y(t)$ is modeled as a function of the time index t [38]. The general form of a polynomial model of degree d is defined as:

$$y(t) = \beta_0 + \beta_1 t + \beta_2 t^2 + \cdots + \beta_d t^d + \epsilon \tag{1}$$

or, more in general:

$$y(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} \beta_{ij} x_i x_j + \cdots + \epsilon \tag{2}$$

where β_i represents the coefficients to be estimated and ϵ denotes the residual error. In this study, we evaluated polynomials of degree 1 (linear), 2 (quadratic), and 3 (cubic) to capture different levels of curvature in the variables' trajectories. The coefficients are estimated using the Ordinary Least Squares (OLS) method, which minimizes the sum of squared differences between observed and predicted values [12]. Such polynomial regression framework was implemented in Python with the `polyfit` from Numpy[22].
While these models are computationally efficient and provide a clear representation of the global trend, they are inherently limited by their rigid functional form. Specifically, polynomial regression lacks the capacity to model local temporal dependencies or adapt to stochastic variations that do not follow a fixed trajectory [8]. Consequently, they serve primarily as a baseline to quantify the performance gain provided by more flexible, non-linear architectures like Long Short-Term Memory (LSTM) networks and Gaussian Process Regression (GPR), which are better suited for the complexities of industrial sensor data [58].

3.4. Gaussian Process Regression

Gaussian Process Regression (GPR) is a Bayesian non–parametric framework[1] that models time series as realizations of stochastic processes characterized by a mean function $m(t)$ and a covariance function $k(t, t')$. Formally,

$$f(t) \sim \mathcal{GP}\big(m(t), k(t, t')\big). \tag{3}$$

The kernel $k(t, t')$ encodes assumptions about smoothness, correlation length, and characteristic temporal structure of the signal. For this reason, GPR is particularly suitable for time series forecasting. Hyperparameters governing the kernel are estimated by maximizing the log marginal likelihood,

$$\log p(\mathbf{y} \mid \mathbf{t}, \theta) = -\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log(2\pi), \tag{4}$$

which automatically balances data fit and model complexity. This property limits overfitting even in the presence of irregular or noisy operating conditions, a well-documented advantage in industrial time series modeling [19].

3.4.1 Kernel Design and Compositional Search Strategy

The expressiveness of a GPR model depends critically on the choice of covariance kernel. In many applications, including the reference framework used in the work by Galeazzi et al. [19], kernels such as the Linear (LIN), Radial Basis Function (RBF), and Rational Quadratic (RQ) serve as building blocks:

$$k_{\mathrm{LIN}}(t, t') = \sigma^2 + t\, t', \tag{5}$$

$$k_{\mathrm{RBF}}(t, t') = \sigma^2 \exp\left(-\frac{(t - t')^2}{2\lambda^2}\right), \tag{6}$$

$$k_{\mathrm{RQ}}(t, t') = \sigma^2 \left(1 + \frac{(t - t')^2}{2\alpha\lambda^2}\right)^{-\alpha}. \tag{7}$$

These kernels capture, respectively, linear trends, smooth nonlinear variations, and multi-scale fluctuations. In these formulations, the hyperparameters govern the structural properties of the functions sampled from the GP prior [47]. The signal variance σ^2 determines the overall amplitude or vertical scale of the fluctuations, effectively scaling the uncertainty of the predictions. In both the RBF and RQ kernels, the characteristic length-scale λ defines the distance in the input space over which the correlation between points decays, thus controlling the smoothness of the underlying function [14]. For the RQ kernel, the shape parameter $\alpha > 0$ dictates a scale mixture of RBF kernels with varying length-scales; as $\alpha \to \infty$, the RQ kernel converges to the RBF, whereas smaller values of α allow the model to capture variations occurring simultaneously across multiple scales [47].

[1] **Bayesian non-parametric framework**: a probabilistic modeling approach in which uncertainty is explicitly represented using probability distributions, and prior assumptions are updated as new data are observed. The term "non-parametric" indicates that the model complexity is not fixed in advance by a finite set of parameters, but can adapt and grow with the amount of available data.

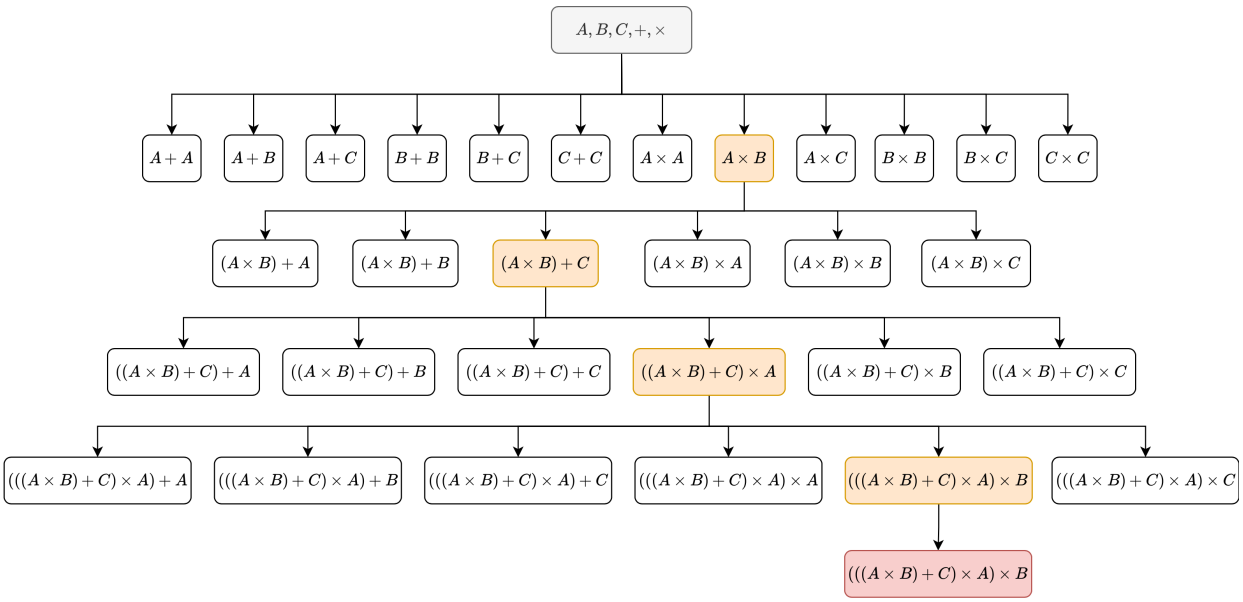


Figure 3: The figure illustrates the hierarchical expansion of a kernel tree, where basic kernels (LIN, RBF, and RQ) are combined using addition and multiplication through a greedy search strategy, highlighting the selected nodes in orange and the final resulting kernel in red[19].

In the specific case of the linear kernel k_{LIN}, the parameter σ^2 represents the variance of the bias (intercept) term, defining the function's uncertainty at the origin.
A common strategy for kernel selection, which was also adopted in the referenced study[19] and used here for comparison with our proposed model, is to construct richer kernels through *compositional combinations* of the base elements using addition and multiplication.
As described by Duvenaud et al. [15], these algebraic operations possess distinct semantic interpretations in the context of modeling. Summation ($k_A + k_B$) models the data as a superposition of independent processes, effectively performing an logical OR operation where the resulting covariance is high if either component is active. This is particularly useful for separating distinct signal components, such as a long-term trend and a seasonal variation. Conversely, multiplication ($k_A \times k_B$) models the interaction between processes, acting as a logical AND operation. This allows for the modulation of structures, such as converting a global periodic structure into a locally periodic one by multiplying a Periodic kernel with an RBF kernel.

To automate this construction, a *greedy tree-based kernel search* is typically performed. Following the methodology detailed in [15] and adapted in [19], the procedure iterates as follows:

1. **Expansion:** The current best-performing kernel is combined with each base kernel (LIN, RBF, RQ) using both addition (+) and multiplication (×) operators.
2. **Pruning:** Redundant combinations (e.g., LIN + LIN) are discarded to maintain model parsimony.
3. **Optimization:** The hyperparameters of new candidates are optimized. To mitigate non-convexity, parameters inherited from the previous iteration are initialized to their optimal values from the parent kernel.
4. **Selection:** Each candidate is evaluated via cross-validated prediction error (e.g., MAE), and the best kernel is selected for the next iteration.
5. **Termination:** The search terminates when a maximum tree depth is reached (e.g., depth of 5) to prevent overfitting and limit computational cost.

The explained procedure can be graphically represented in Figure 3.
Although not globally optimal, this search procedure efficiently explores a broad space of kernel structures while remaining computationally tractable. It provides a systematic and reproducible strategy for identifying expressive kernels, and it is therefore used as a benchmark against which we compare the performance of our proposed modeling approach. As already discussed, a detailed description of this methodology appears in the referenced work, which we adopt as a baseline scheme [19]. This scheme was implemented to search the optimal kernels for the two case studies considered by this work; the Scikit-learn[42] library was exploited to train the GPR models.

3.5. Recurrent Neural Networks

Recurrent Neural Networks (RNNs), first proposed in 1990 (Elman, 1990 [16]), represent a fundamental paradigm shift in neural network design, moving from static to dynamic information processing. While traditional feed-forward neural networks (FFNNs) are designed to map inputs to outputs through a unidirectional, acyclic flow, they inherently operate under the assumption that each data point is independent and identically distributed [50]. This architecture effectively treats each observation as an isolated event, which poses a significant limitation when dealing with sequences where the temporal order is not merely an attribute but a core feature of the data's underlying structure.

In contrast to these "stateless" models, RNNs are specifically engineered to handle sequential data by introducing a mechanism for internal persistence. In complex industrial environments, such as the oil regeneration plant considered in this study, the physical state of a system is never truly instantaneous. For instance, the pressure drop or the temperature profile within a furnace at a given time t is the result of a continuous evolution governed by chemical and physical kinetics occurring at times $t-1, t-2$, and so on. RNNs address this by incorporating recurrent connections, which allow information to circulate within the network through feedback loops.

This recursive structure transforms the network into a dynamic system capable of maintaining an internal "hidden state." This state acts as a form of short-term memory that captures a compressed representation of the sequence's history. By updating this memory at every time step, the network does not simply process the current input in isolation; instead, it interprets the current signal in the context of the information it has accumulated from the past.

Consequently, RNNs are uniquely suited for modeling time-series where the detection of trends, cycles, and long-term dependencies is crucial. By effectively "remembering" the trajectory of the input data, these networks can distinguish between similar instantaneous observations that may have vastly different implications based on their preceding context. This ability to integrate historical context into current computations allows for a more nuanced understanding of non-linear dynamical systems, making RNNs a powerful tool for monitoring and predicting the performance of industrial equipment.

Recursive vs. Multi-step Forecast Strategy

A crucial design choice in time-series forecasting concerns how predictions are generated across a multi-step horizon. Traditional *recursive* forecasting strategies, often referred to as iterative or rolling-horizon approaches, predict one time step ahead and then feed that specific prediction back into the model as an input to infer subsequent steps. While this approach is computationally efficient and allows for an indefinite prediction horizon using a single model, it inherently suffers from the *Compounding-Error Problem* [3]. In such a setup, any small divergence from the ground truth at the first predicted step is amplified in all following steps, as the model is forced to operate on its own potentially noisy estimates rather than real observations. This phenomenon, often termed *Exposure Bias*[48], leads to a rapid degradation of the predictive trajectory, which can cause the model to drift toward unrealistic or unstable states when forecasting far into the future.

To explicitly avoid this error propagation, this work employs a *direct multi-step forecasting* strategy. In this configuration, the model is designed to map a high-dimensional input window of length L directly to a high-dimensional output vector representing the entire future horizon of length H in a single forward pass:

$$\hat{\mathbf{y}}_{t+1:t+H} = f_\theta(x_{t-L+1:t}). \tag{8}$$

When learning the joint temporal structure of all H future steps simultaneously, the model is optimized to maintain consistency across the forecasting horizon, which is essential for accurate **long-term forecasting**. From a learning perspective, the direct strategy effectively treats the forecasting task as a multi-output regression problem, allowing the model to capture the overall shape of the degradation curve—such as the accelerating trend of the pressure drop due to coking—rather than focusing on local step-to-step variations. This approach provides a much more robust global forecast for the industrial furnace, where the stability of the remaining useful life estimate is paramount and must remain unaffected by the cumulative volatility and divergence typical of recursive models.

3.5.1 Loss function and Backpropagation Through Time

Let $\mathbf{X}_t \in \mathbb{R}^{n\times d}$ denote the input at time step t and $\mathbf{H}_t \in \mathbb{R}^{n\times h}$ the corresponding hidden state, where n is the number of samples, d the input dimensionality, and h the number of hidden units. The hidden state is updated according to

$$\mathbf{H}_t = \phi_h\left(\mathbf{X}_t\mathbf{W}_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h\right), \tag{9}$$

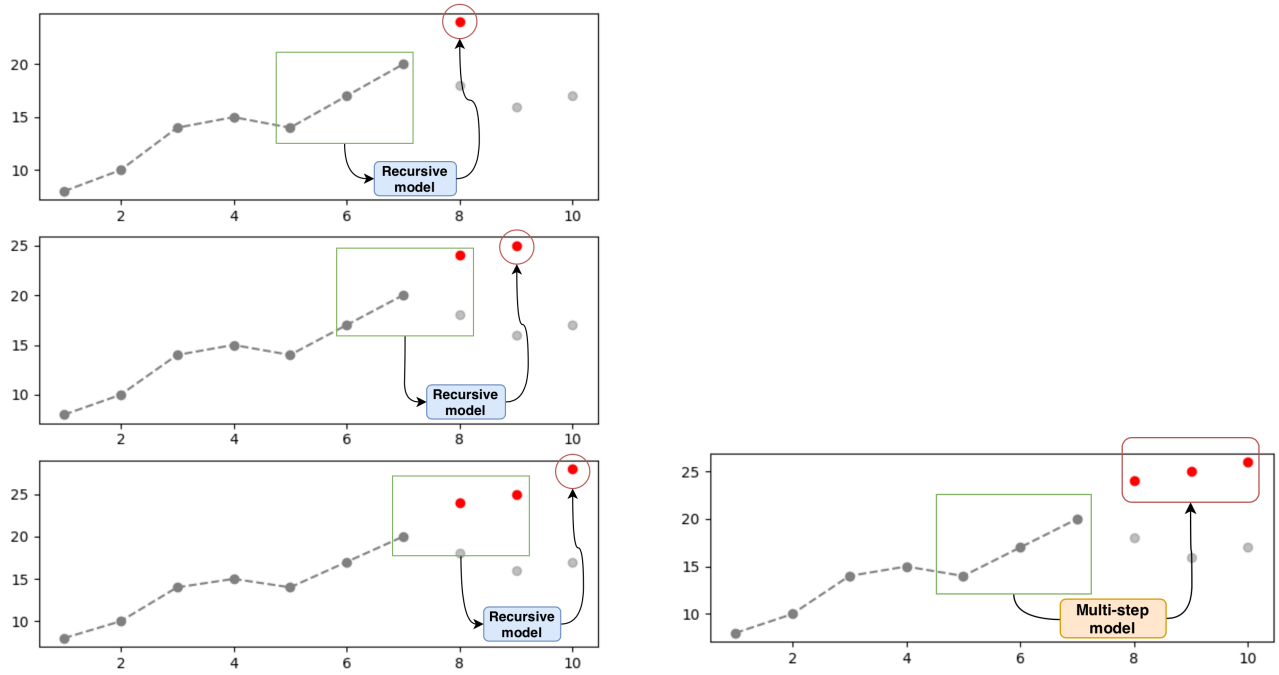


Figure 4: Conceptual comparison between recursive prediction (on the left) and multi-step prediction (on the right).

where $\mathbf{W}_{xh} \in \mathbb{R}^{d\times h}$ is the input-to-hidden weight matrix, $\mathbf{W}_{hh} \in \mathbb{R}^{h\times h}$ the recurrent weight matrix, $\mathbf{b}_h \in \mathbb{R}^{1\times h}$ a bias term, and ϕ_h a nonlinear activation function (typically tanh or sigmoid). The output at time t is given by

$$\mathbf{O}_t = \phi_o\left(\mathbf{H}_t\mathbf{W}_{ho} + \mathbf{b}_o\right), \tag{10}$$

with $\mathbf{W}_{ho} \in \mathbb{R}^{h\times o}$ and output activation ϕ_o. The recursive dependence of $\mathbf{H}_t$ on $\mathbf{H}_{t-1}$ implies that the hidden state at time t encodes information from all previous time steps.

Training an RNN requires computing gradients of the loss with respect to all parameters across time. Given a sequence of target outputs $\mathbf{Y}_t$, the total loss over a sequence of length T is defined as

$$\mathcal{L}(\mathbf{O}, \mathbf{Y}) = \sum_{t=1}^{T} \ell_t(\mathbf{O}_t, \mathbf{Y}_t), \tag{11}$$

where ℓ_t is a task-specific loss function (e.g., mean squared error or cross-entropy).
Backpropagation Through Time (BPTT) computes gradients by unrolling the RNN over time and applying the chain rule. For the output weights, the gradient is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ho}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t}\frac{\partial \mathbf{O}_t}{\partial \phi_o}\mathbf{H}_t. \tag{12}$$

For the recurrent and input weight matrices, the temporal dependencies introduce additional terms. The gradients can be written as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t}\frac{\partial \mathbf{O}_t}{\partial \phi_o}\mathbf{W}_{ho}\sum_{k=1}^{t}\frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k}\frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{hh}}, \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t}\frac{\partial \mathbf{O}_t}{\partial \phi_o}\mathbf{W}_{ho}\sum_{k=1}^{t}\frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k}\frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{xh}}. \tag{14}$$

Using the recursive structure of the hidden state, these expressions can be rewritten as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t}\frac{\partial \mathbf{O}_t}{\partial \phi_o}\mathbf{W}_{ho}\sum_{k=1}^{t}\left(\mathbf{W}_{hh}^{\top}\right)^{t-k}\mathbf{H}_k, \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t}\frac{\partial \mathbf{O}_t}{\partial \phi_o}\mathbf{W}_{ho}\sum_{k=1}^{t}\left(\mathbf{W}_{hh}^{\top}\right)^{t-k}\mathbf{X}_k. \tag{16}$$

3.5.2 Vanishing and exploding gradients

The presence of powers of $\mathbf{W}_{hh}$ in the gradient expressions highlights a fundamental issue of BPTT: gradients may vanish or explode depending on the eigenvalues of the recurrent weight matrix. Eigenvalues with magnitude smaller than one cause gradients to decay, while eigenvalues larger than one lead to divergence (Bengio et al., 1994[4]). Currently, scientists agree on two possible different techniques to get around this critical aspect: *Truncated Backpropagation Through Time* (TBPTT) and *Long Short Term Memory* (LSTM) cells (Schimdt, 2019[49]).

Truncated Backpropagation Through Time (TBPTT) addresses this problem by limiting the backward flow of gradients to a fixed number of time steps. Practically, this truncation imposes a sliding window over the unfolded computational graph, restricting how far into the past error signals can propagate. This reduces computational cost and improves numerical stability, at the expense of limiting the effective temporal dependency length that the RNN can learn[49].

Likewise, LSTM cells (Hochreiter, Schmidhuber, 1997[23]) are introduced to solve the vanishing/exploding gradient problem, but they do it by storing information within time steps. The behavior of such units is governed by an internal memory state, called the *cell state*, and three control mechanisms called *gates*, which regulate the flow of information over time. Given an input x_t, the previous hidden state h_{t-1}, and the previous cell state C_{t-1}, the LSTM cell operates as follows.
The **forget gate** determines which information from the previous memory should be retained:

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$$

The **input gate** decides which new information should be added to the cell state:

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_i\right)$$

Simultaneously, a candidate update for the cell state is computed:

$$\tilde{C}_t = \tanh\left(W_c[h_{t-1}, x_t] + b_c\right)$$

The cell state is then updated by combining retained and new information:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Finally, the **output gate** determines which part of the updated cell state contributes to the current hidden state:

$$o_t = \sigma\left(W_o[h_{t-1}, x_t] + b_o\right)$$
$$h_t = o_t \odot \tanh(C_t)$$

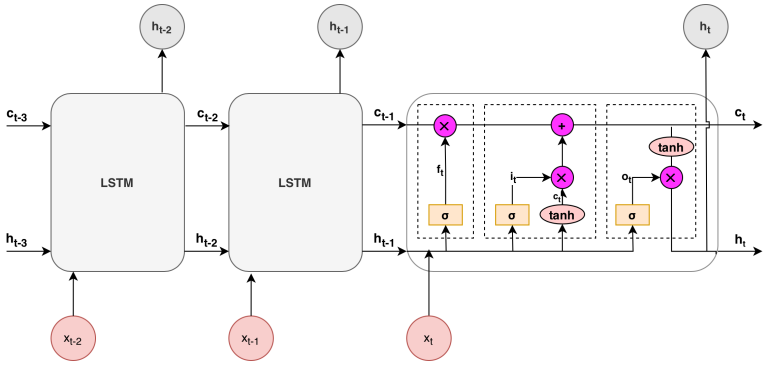


Figure 5: Conceptual scheme of a LSTM cell [37].

3.6. Proposed LSTM-Based Architecture

Building upon the theoretical foundations outlined in Section 3.5, we now describe the specific LSTM-based architecture adopted in this work for direct multi-step forecasting. The model is designed to jointly capture short-, medium-, and long-range temporal dependencies through a stacked recurrent structure equipped with

dropout regularization and a horizon-aware output layer.

The architecture consists of two sequential LSTM layers. The first LSTM layer is configured to return the full sequence of hidden states, thereby allowing the model to process and transform temporal information at every step within the input window. This design enables the extraction of enriched representations of the underlying temporal patterns, which are then passed to the second recurrent layer. Wile the first recurrent layer primarily captures low-level, short-term dependencies in the input sequence, while the second layer operates on these hidden representations to model higher-level, longer-term temporal patterns and nonlinear interactions. This layered abstraction produces a more informative latent state that is better aligned with the forecasting objective, allowing the fully connected layer to more effectively map historical information to multiple future time steps. As a result, stacked RNNs often improve forecast accuracy for complex time series, particularly when predicting longer horizons (Mienye, Swart, 2024[36]).

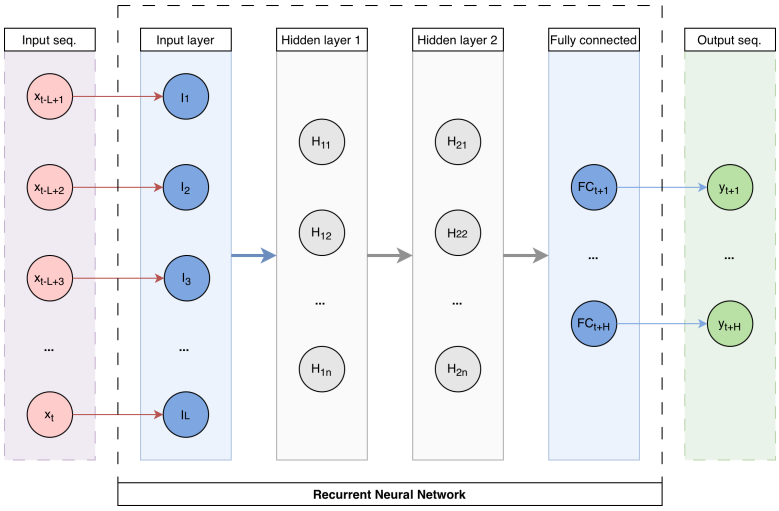


Figure 6: General architecture of an RNN

To enhance the model's generalization capability, each LSTM layer is followed by a dropout mechanism. During training, dropout randomly deactivates a fraction of the recurrent units, effectively reducing co-adaptation among neurons and encouraging the network to learn more robust, distributed representations. This regularization is particularly important in time-series forecasting, where models otherwise tend to overfit to local temporal fluctuations rather than learn persistent long-range structures (Zaremba et al., 2015[55]).

The final stage of the architecture is a fully connected (dense) output layer whose dimensionality corresponds to the forecasting horizon H. Unlike recursive one-step-ahead schemes, where predictions are sequentially fed back into the model, this dense layer directly maps the latent temporal representation into a vector of H predicted future values. This design enforces the simultaneous learning of all temporal dependencies across the horizon, improving long-range consistency and stability in the forecasts.

Training follows the direct multi-step paradigm described earlier: given an input window of length L, the model outputs the entire H-step forecast in a single forward pass. Optimization is carried out using the Adam algorithm and the mean squared error loss, computed over all future time steps of the prediction horizon. With this process, the model is forced to internalize temporal patterns not only within the input window but also across the predicted output sequence, leading to more coherent and physically plausible forecasts.

RNN selected architecture

	Size	**Dropout %**
Recurrent layer 1	50 LSTM units	20
Recurrent layer 2	50 LSTM units	20
Fully connected layer	H neurons	-

3.7. Performance Metrics

To evaluate the performance of the models, three standard metrics were employed: the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE), and the coefficient of determination R^2. The MAE quantifies the average absolute difference between the predicted and observed values,

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

and is particularly useful when an interpretable metric that is less sensitive to outliers is desired. The RMSE, defined as

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

penalizes larger errors more strongly and is therefore preferable when the emphasis is on accurately capturing substantial deviations. To compare the performance of the model across different variables and case studies, it was chosen to use the relative RMSE, thus:

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

3.8. Data Augmentation

A significant challenge when researching on industrial time series forecasting is the scarcity of publicly available data. In particular, we faced this issue with industrial furnace case study, as only a single operational run was accessible for training. To address this, we implemented a data augmentation pipeline inspired by the principle of "guided" transformations proposed by Iwana and Uchida [28].
While Iwana and Uchida use existing data to "guide" the deformation of time series, our approach adapts this philosophy to a single-run scenario by using domain-specific heuristics to guide the generation of realistic variants. This ensures that the synthetic data preserves the physical consistency of the process rather than relying on purely random noise. We generated $K = 10$ synthetic "pseudo-runs" by applying the following transformations:

1. **Baseline drift simulation:** To account for sensor aging or thermal fluctuations, we introduced a low-frequency drift ϵ_{low}, mimicking the long-term trends often observed in industrial sensors:

$$\epsilon_{low} = G_{\sigma=20}(\mathcal{N}(0, 2)) \tag{17}$$

2. **Dynamic scaling and periodic interference:** We simulated variations in signal intensity and electrical noise using a stochastic stretching factor $s_t \sim \mathcal{U}(0.95, 1.05)$ and a sinusoidal modulation m_t:

$$m_t = 1 + 0.02 \cdot \sin(2\pi f t) \tag{18}$$

3. **Simulated shifts:** To replicate sudden mechanical shocks or recalibrations, we injected $N \in [3, 7]$ discrete jumps (J_t), teaching the model to remain robust against abrupt offsets:

$$J_t = \sum_{p<t} a_p, \quad a_p \sim \pm\mathcal{U}(0.1, 0.5) \tag{19}$$

4. **Non-linear refinement:** The augmented signal $X' = (X \cdot s_t \cdot m_t) + \epsilon_{low} + J_t$ was processed via a power-law transformation and Gaussian smoothing ($\sigma = 2$). Similar to the use of shape-descriptors in guided warping, this step ensures that the local "shape" of the signal remains smooth and physically plausible:

$$X_{final} = G_{\sigma=2}\left(\text{sgn}(X') \cdot |X'|^{1.05}\right) \tag{20}$$

The consistency in performance across both the augmented furnace data and real operational cycles from the batch fermentation dataset suggests that this "guided" heuristic approach effectively captures the underlying variance required for robust modeling.

3.9. Experimental Design and Study Protocol

To rigorously evaluate the effectiveness of the proposed forecasting framework, the experimental campaign is structured into a hierarchical protocol consisting of three distinct sequential phases. This approach is designed to progressively isolate the challenges of industrial time-series forecasting and to demonstrate how the proposed methodology overcomes them. The experimental logic follows the breakdown reported in Table 1.

Phase I: Probabilistic benchmarking (single-run) The first phase establishes a performance baseline using conventional modeling techniques on a single operational run. In this stage, Polynomial Regressions (PR) are employed to characterize global deterministic trends, while Gaussian Process Regression (GPR) serves as the state-of-the-art probabilistic benchmark[19]. The primary objective of this phase is to document the limitations of standard regression models, specifically their inability to capture high-frequency stochastic fluctuations and their tendency to miss non-stationarity over long forecasting horizons. This phase is applied to both the *Industrial Furnace* and the *Batch Fermentation* case studies to assess the performance of traditional methods over complex time series data.

Phase II: Assessment of model instability (limited data scenario) The second phase addresses the critical issue of applying Deep Learning architectures in data-scarce environments. Focusing specifically on the *Industrial Furnace* case, where historical data is initially limited to a single fouling cycle, an LSTM network is trained, with the architecture described in Section 3.6. This "stress test" is crucial to demonstrate that deep learning models are not inherently robust; when forced to learn from a single sequence, they are highly susceptible to overfitting and local noise memorization. This concept is often summarized by the principle known as "garbage in, garbage out," originally attributed to George Fuechsel[18], which states that poor-quality input data inevitably leads to poor-quality model outputs. This phase studies the performance of Deep Learning architectures in limited data scenarios and the need for additional data.

Phase III: Comparative evaluation on multi-run datasets The third phase considers a scenario with abundant data for the Deep Learning model training. Since the furnace dataset only contains one run originally, data augmentation was applied to synthetically create other similar runs, resulting in a more diverse dataset. Therefore, while the *Batch Fermentation* case utilizes the original dataset of 100 runs, the *Industrial Furnace* case employs an augmented dataset including 10 synthetic runs generated to replicate physical consistency in addition to the original one. The proposed model is compared against the GPR baseline across two forecasting horizons, H_{short} (0.25τ) and H_{long} (0.5τ), as discussed in Section 3.2. Performance is assessed through a multi-metric approach, combining qualitative trajectory inspection, analysis of loss convergence, and quantitative evaluation using the relative Root Mean Square Error (rRMSE).

Table 1: **Study protocol**

	Industrial Furnace Case	**Batch Fermentation Case**
Probabilistic Benchmarking	Single-run	Single-run
Limited Data Scenario	Single-run	-
Comparative Evaluation	11 runs (augmented)	100 runs (fully original)

Table 1: Summary of the hierarchical experimental protocol, detailing the data volume and objectives for each phase of the study.

4. PREDator: A Dedicated Tool for Predictive Modeling

To facilitate the experimental phase and manage the complexity of recurrent neural network architectures, a specialized Graphical User Interface (GUI) named `PREDator` was developed in Python. The primary rationale behind the creation of this tool is to bridge the gap between high-level algorithmic design and practical model deployment, providing a standardized environment for training and testing temporal models on industrial datasets.

The development of PREDator addresses several fundamental requirements of the research workflow. First, it ensures experimental consistency by abstracting the model's configuration. Through the interface, critical parameters such as the input rolling window (L), the forecast horizon (H), and specific network hyperparameters can be adjusted without direct modification of the underlying source code. This minimizes the risk of human error during iterative testing and allows for a more systematic exploration of the model's sensitivity across various industrial data streams.

Second, the tool automates the data-handling and modeling pipeline. Upon loading a dataset, PREDator handles the preprocessing, training, and validation phases within a unified framework, enabling rapid iteration and performance comparison across different time-series sequences. A key strength of the software lies in its architectural modularity: while it features a native, pre-implemented package for Gaussian Process Regression

(GPR), it is designed to be model-agnostic. This allows researchers to develop and test arbitrary RNN architectures, integrating new neural structures into the PREDator execution engine by modifying at most a single line of code.
Finally, a core feature of the tool is its automated result traceability. Recognizing the importance of rigorous documentation in data science workflows, PREDator is designed to autonomously generate a structured output repository for every execution. This repository acts as a comprehensive "digital snapshot" of the experiment, containing:

- The serialized model weights for future deployment or fine-tuning (`.h5` format);
- High-resolution graphical representations of loss convergence and forecasting accuracy;
- Comprehensive `.csv` logs documenting performance metrics, hyperparameter settings, and prediction results.

Using these functions in one central place, PREDator transforms the training process from a manual script-based task into a robust, reproducible, and scalable analytical pipeline, specifically designed for processing industrial time series of data. The full codebase is released on GitHub: `https://github.com/antoniomazzitelli/PREDator.git`.

Figure 7: PREDator Logo

4.1. Architecture and Implementation

The architecture of PREDator is designed following a modular approach that ensures a clean separation between the user interface and the underlying computational backend. Developed entirely in Python, the tool leverages the Tkinter[33] framework for its frontend, providing a lightweight and responsive environment for parameter configuration and process control. To maintain application stability during the computationally intensive training phases of the Recurrent Neural Network, the software implements an asynchronous execution strategy via the `threading` module[53]. This multithreading approach offloads the training pipeline to a secondary thread, preventing the graphical interface from freezing and allowing for a seamless user experience while the model processes complex industrial datasets.
The core of the analytical engine is primarily built upon the TensorFlow[1] and Keras[10] libraries, which handle the construction, optimization, and evaluation of the RNN models. Complementing the deep learning backbone, the architecture explicitly includes a dedicated package for Gaussian Process Regression (GPR). Implemented to offer a probabilistic alternative to the LSTM network, this module allows users to select GPR as the predictive engine, thereby facilitating comparative analysis between recurrent neural approaches and non-parametric kernel methods. PREDator manages the entire data lifecycle internally: once the raw industrial time-series are loaded, the tool executes an automated preprocessing pipeline that includes feature scaling and the generation of the rolling window structures necessary for temporal learning. The transition from configuration to execution is handled by a centralized controller function, triggered by the user interface, which orchestrates the flow of data from the ingestion layer to the selected model.
Regarding data persistence and traceability, the tool implements a systematic I/O management logic using the Python `os` module. Within a user-defined root directory, PREDator programmatically generates structured subfolders for each individual simulation run. This hierarchical organization ensures that all experimental artifacts, including serialized model weights, performance logs in `.csv` format, and high-resolution visualization plots, are preserved in a consistent and easily accessible format, facilitating the comparative analysis of different forecasting horizons and network configurations.

4.2. User Interface and Operational Workflow

The PREDator interface is characterized by a user-centric design that prioritizes operational clarity and experimental rigor. The layout is structured to guide the user through a linear workflow, starting from data ingestion to the final generation of results. The interface features dedicated control elements for file management, including specific buttons for dataset selection and output directory definition. Using standard OS-native file dialogs, the tool ensures high flexibility, allowing the user to easily navigate the local file system to load industrial CSV

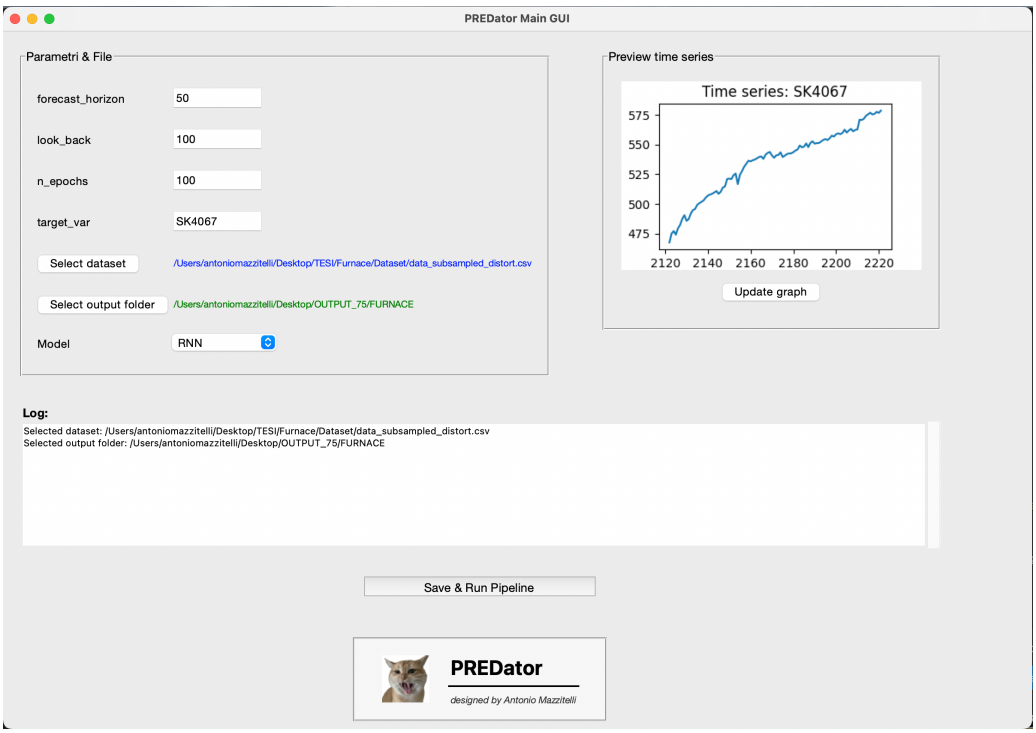


Figure 8: PREDator main GUI

datasets and specify the destination for experimental artifacts.
Central to the GUI is the configuration panel, where the user can define the fundamental temporal and structural parameters of the simulation. Specifically, the tool allows for the adjustment of the input rolling window length (L), the target forecast horizon (H), and the number of training epochs. To maintain the robustness of the experimental setup, PREDator incorporates an input validation layer; any syntactical or logical inconsistencies in the parameter fields, such as non-numeric entries or out-of-bounds values, trigger immediate feedback through dedicated warning message boxes. This prevents the execution of flawed training routines and ensures the integrity of the modeling process.
During the computational phase, the interface provides real-time monitoring of the model's progress. Instead of a silent execution, PREDator features an integrated terminal-like console that streams the training logs directly from the backend. This allows the operator to track the loss convergence and the advancement of epochs (or fitting iterations) in real-time, providing immediate visibility into the network's learning behavior. Once the process is concluded, the tool follows a "save-and-notify" logic: all graphical outputs, including accuracy plots and performance metrics, are silently exported to the previously selected directory. The user can then access the structured subfolders for post-experimental analysis, ensuring that the GUI remains uncluttered while preserving a high degree of detail for the final technical assessment.

Folder	Description
forecasts	Contains all forecasting outputs, including per-epoch performance metrics, generated prediction files, and the final trained RNN–LSTM model.
logs	Stores runtime logs such as training progress, warnings, errors, and system messages.
gpr	Includes forecasting results produced by the Gaussian Process Regression (GPR) module, along with associated performance metrics.

Table 2: Overview of the folders automatically generated inside the selected output directory.

5. Tools

The computational framework was developed using the Python 3 programming language within the Visual Studio Code integrated development environment. The core of the predictive model relies on the TensorFlow ecosystem, specifically using the Keras API for the design and implementation of the LSTM layers.

5.1. Software Stack

The computational framework was implemented using the Python programming language (v. 3.12) [53]. Management of system paths and directory structures was handled via the Python standard `os` library.
Data manipulation and dataset management were performed using Pandas [35], while NumPy [22] was employed for efficient vectorization and numerical operations.
For statistical modeling and preprocessing, the Scikit-learn library [42] was utilized; specifically, it provided the implementation for Gaussian Process Regression (GPR), utilities for data scaling (Scaler), and functions for computing performance metrics. Deep learning architectures and neural networks were developed and trained using the TensorFlow framework [1] exploiting the Keras API [10]. Finally, all graphical visualizations were generated via Matplotlib [25]. Moreover, the `PREDator` GUI was created with the Tkinter toolbox [33].

5.2. Hardware Configuration

To ensure the reproducibility of the results, the hardware specifications used for training and testing the model are summarized in Table 3.

Category	Specification
Hardware Platform	MacBook Air (2020), 1.1 GHz Quad-Core Intel Core i5, 8GB RAM
Operating System	macOS
Development Environment	Visual Studio Code (VS Code)
Programming Language	Python 3.12
Deep Learning Framework	TensorFlow 2 w/ Keras API
Data Handling	Pandas, NumPy, Scikit-learn
Visualization	Matplotlib

Table 3: Summary of the computational environment, hardware specifications, and software libraries utilized for the model development and experimental phase.

6. Results

This chapter presents the experimental validation of the proposed framework applied to the selected industrial case studies. The analysis begins by evaluating the limitations of standard probabilistic and deterministic baselines in capturing long-term dependencies (§6.1). Subsequently, the instability of recurrent architectures in data-scarce scenarios is assessed(§6.2). Finally, the performance of the LSTM-based framework is benchmarked against these baselines, demonstrating its superior robustness and accuracy in multi-step forecasting.

6.1. Probabilistic Benchmarking via Gaussian Process Regression

The qualitative analysis of the Gaussian Process Regression (GPR) baseline reveals how the model's performance is strictly linked to the length of the forecast horizon. In the short term, the GPR demonstrates a good ability to follow local trends, providing a reliable probabilistic estimate. However, as the horizon increases, we observe a progressive loss of predictive accuracy, as evident in Figure 9c vs 9d. This divergence indicates that the model, while robust for near-future steps, struggles to capture the long-term temporal dependencies required for complex industrial processes.

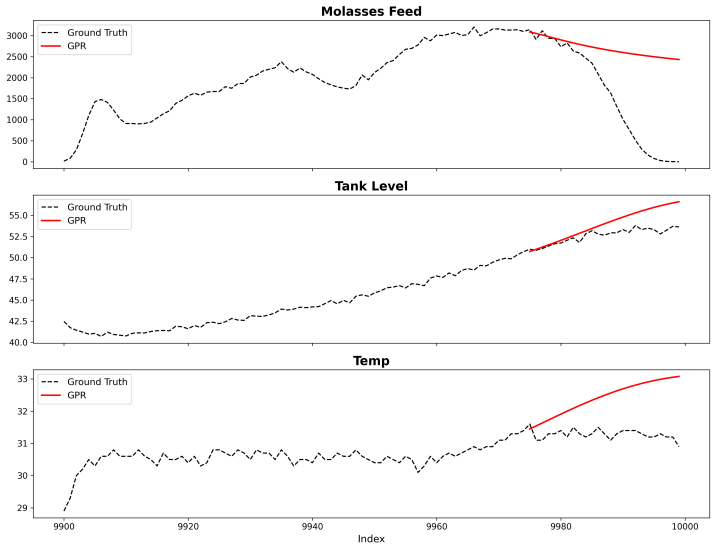
As discussed in Section 3.4.1, the kernel selection follows the greedy tree strategy established by Galeazzi et al. [19]. This method provides a structured and reproducible way to build the model; however, it tends to favor kernels that produce smooth extrapolations. While this "smoothness" is useful to avoid overfitting on noisy data, it acts as a limitation when the model must predict sharp transitions, such as the onset of fouling in a furnace or the phase changes in batch fermentation. In these scenarios, the GPR often produces conservative, mean-reverting predictions that fail to track the actual fluctuations and non-linearities of the process.
This failure is most evident in the furnace case study, specifically regarding the long-term prediction of the pressure drop (PI4301). The pressure drop is a critical indicator of fouling and exhibits a distinct, non-linear upward trend as the cycle progresses. As shown in Figure 9d, the GPR fails to sustain this trajectory over extended horizons, eventually reverting toward the historical mean and significantly underestimating the final values. This inability to track the most critical variable for maintenance scheduling underscores the structural limitations of the GPR when dealing with evolving industrial dynamics that deviate from a steady-state or a simple periodic behavior.
These observations highlight that while the GPR is a valid baseline, its performance is highly sensitive to the static nature of the validation set used for kernel tuning. The model effectively "learns" a general trend but lacks the flexibility to adapt to shifting operational conditions over long periods. To improve this benchmark, future work could implement a rolling validation procedure. By testing the kernel performance over successive moving windows, it would be possible to select configurations that prioritize long-term stability, providing a more challenging and realistic comparison for deep learning architectures.
Furthermore, an attempt was made to improve the model's generalization by training the GPR on multiple historical cycles rather than a single run. However, this approach highlighted the significant computational limitations inherent in Gaussian Processes. Since the training process requires the inversion of the covariance matrix, the computational complexity scales cubically with the number of data points, denoted as $O(N^3)$[19]. As the training set expands to include even a few industrial batches, the memory requirements and processing time increase exponentially, making the training phase extremely demanding for standard industrial hardware. Despite this significant increase in computational effort, the inclusion of multiple cycles did not lead to a substantial improvement in forecasting accuracy. The model still struggled to distinguish between cycle-specific noise and the underlying process dynamics, confirming that simply increasing the volume of data is not sufficient to overcome the GPR's structural limitations in long-term sequence modeling.

6.1.1 Polynomial Regression on single-run data

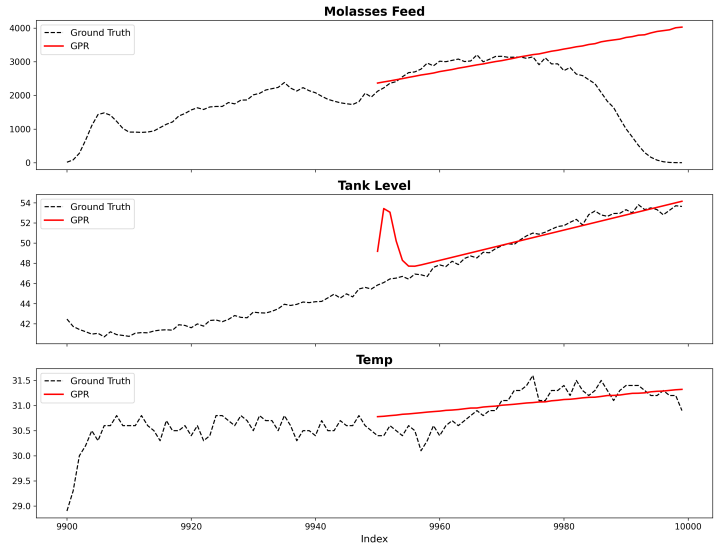
To include a deterministic baseline, Polynomial Regression was performed on the Furnace dataset. As illustrated in Figure 10, provide a qualitative understanding of the variable's global trajectory but highlight significant limitations regarding forecasting accuracy.
At a first glance, the polynomial models appear to capture the general upward trend of the variables (FI4091, PI4301, and SK4067). However, a closer inspection reveals their inability to track the high-frequency oscillations and the inherent noise of the physical process or steep decreases as happens during blowing operations. While these models provide a smooth approximation of the global growth, they fail to recognize complex temporal patterns and local sequences that characterize the sensors' behavior.
In the context of Predictive Maintenance, or Predictive Modeling in general, these shortcomings are critical. The rigid nature of polynomial functions prevents the model from capturing intrinsic deviations or non-linear trends that often serve as early indicators of fouling, malfunctioning and degradation. Furthermore, as shown in the forecast region (post-training split), higher-degree polynomials (e.g., degree 3 for PI4301) exhibit poor generalization, often diverging significantly from the actual data path. This lack of sensitivity to sequential dependencies and stochastic fluctuations confirms that simple deterministic models are insufficient for robust predictive maintenance for instance, necessitating the transition toward more sophisticated architectures like recurrent neural networks, which are intrinsically capable of managing time series or, more in general, sequential arrays of data.

6.2. Assessing Model Instability in Limited Data Scenarios

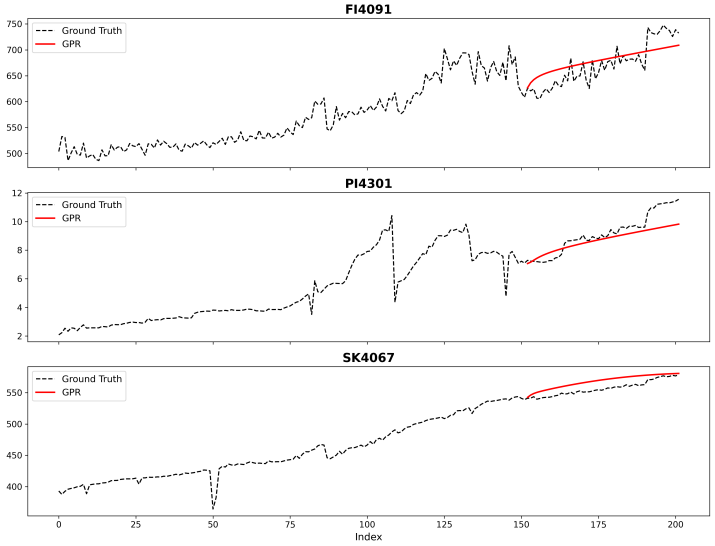
Prior to the definitive validation of the PREDator framework, it is essential to evaluate the performance of recurrent architectures when subjected to the typical data constraints of industrial environments. This phase focuses on the industrial furnace case study, and only single-run time series were used to train the network. As detailed in the methodology, an RNN was trained on this highly constrained dataset to assess its capacity to capture the temporal dynamics of the FI4091, PI4301, and SK4067 sensors without the support of data enrichment.
The results, illustrated in Figure 11, reveal the significant challenges associated with training deep learning



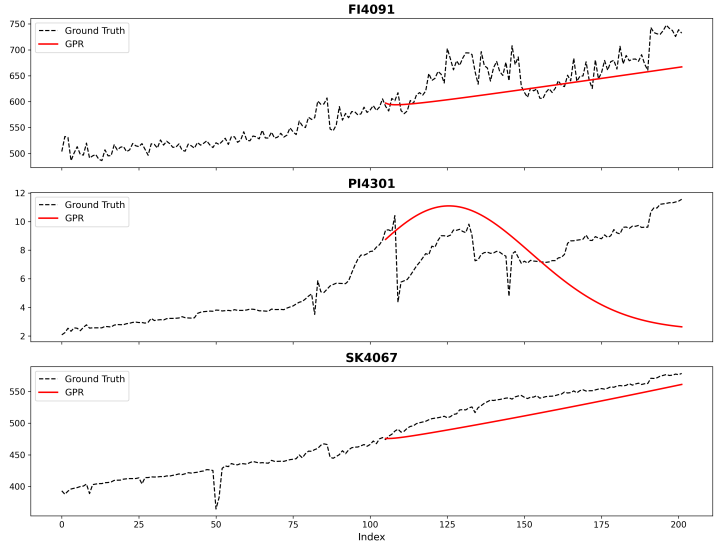
(a) Batch fermentation – short horizon



(b) Batch fermentation – long horizon



(c) Industrial furnace – short horizon



(d) Industrial furnace – long horizon

Figure 9: Forecasts with long and short horizons, performed with Gaussian Process Regression on single-run time series of data

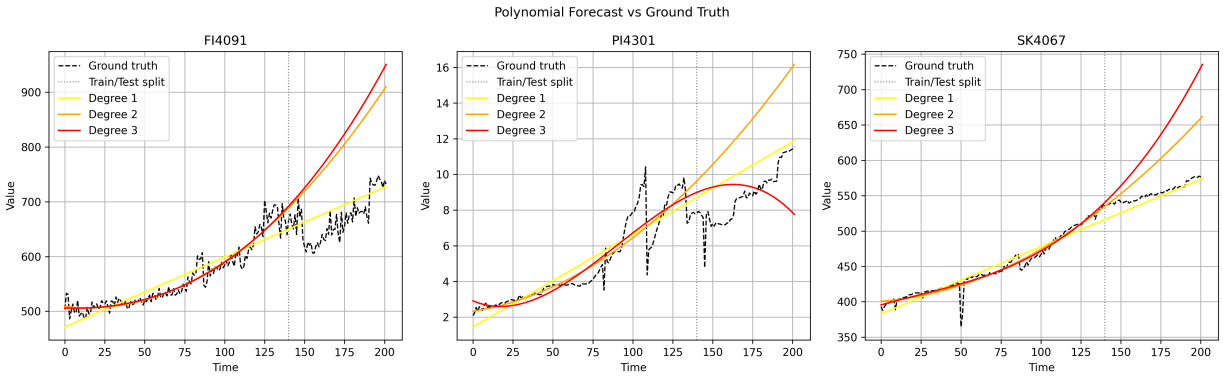


Figure 10: Polynomial regression of the three considered variables of the industrial furnace.

models in data-scarce environments. Despite evaluating the model across 20, 50, and 100 training epochs to observe potential convergence, the network fails to provide a stable or accurate representation of the process. Even as the number of epochs increases, the model does not refine its predictive accuracy; instead, it exhibits a persistent forecasting bias and an inability to generalize the underlying physical phenomena.

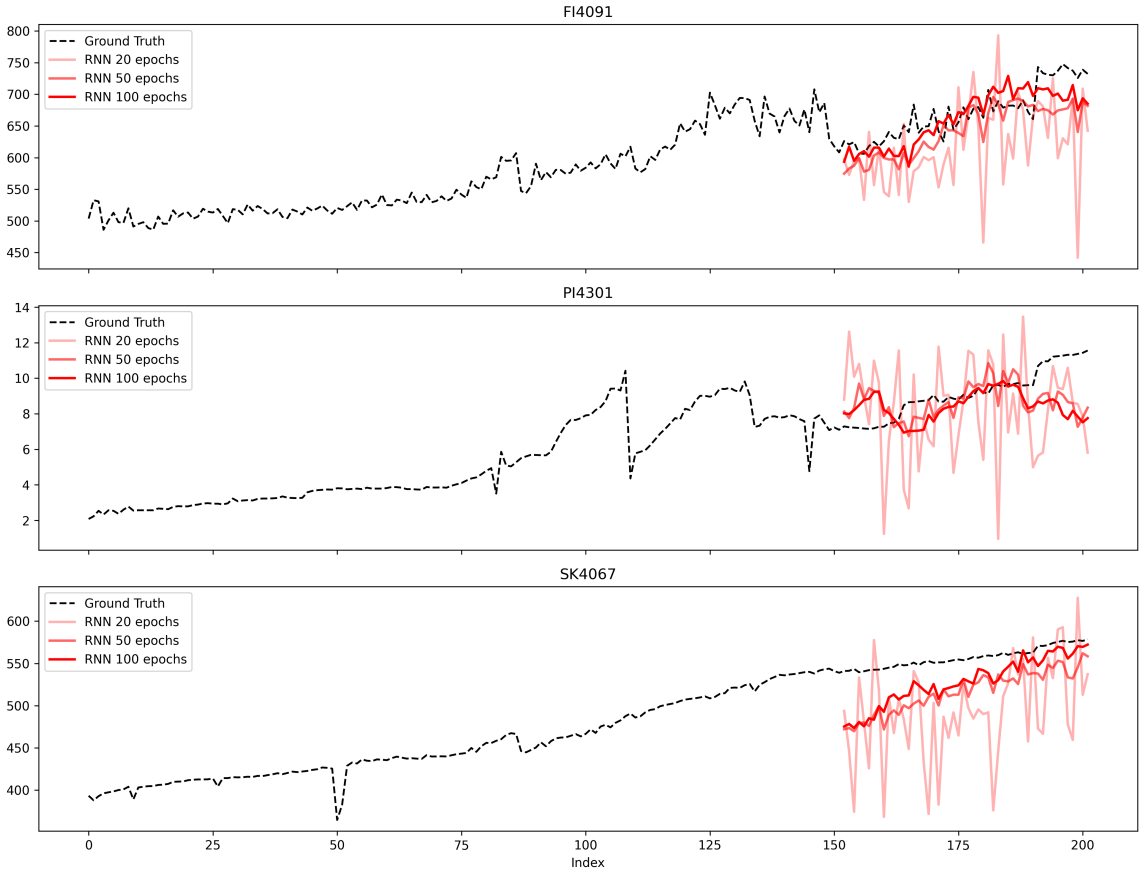


Figure 11: Evaluation of the RNN architecture on a single-run dataset, trained with 20, 50, and 100 epochs.

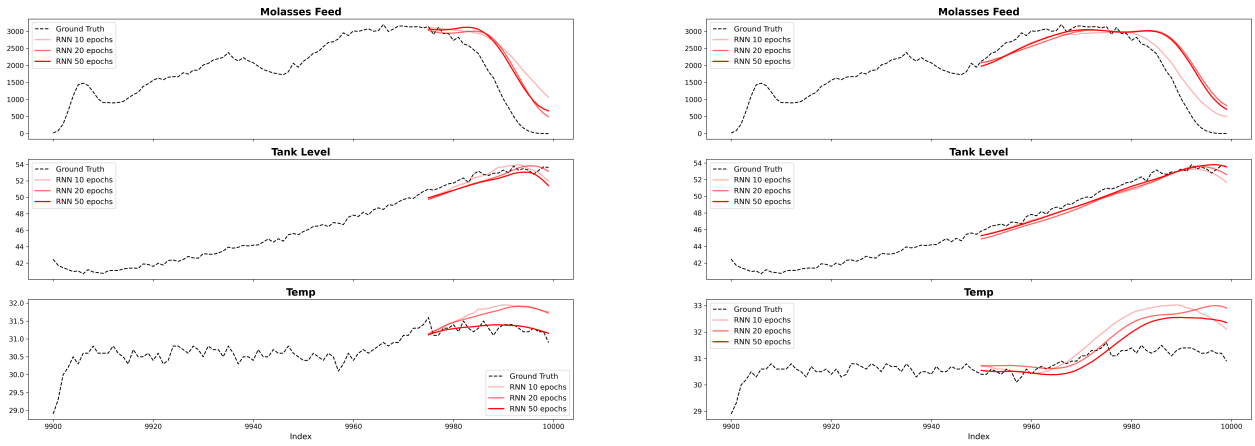
The poor predictive performance of the architecture is rooted in the complex, non-stationary nature of the furnace's operation. Throughout a single run, the process undergoes a continuous evolution driven by fouling, which progressively shifts the baseline operating conditions and alters the signal dynamics. A model trained on a single cycle lacks the statistical perspective required to distinguish between transient stochastic noise and the long-term deterministic drift caused by equipment degradation. Consequently, the features learned during the initial training phase become increasingly unrepresentative of the system's state as the run progresses, leading to a total loss of predictive reliability in the terminal part of the sequences.
This instability is further exacerbated by the operational profile of the furnace, which includes discrete, high-

impact interventions such as steam blowing operations. These events are clearly visible in the PI4301 signal as sharp, sudden drops in pressure intended to mitigate fouling effects. Because the RNN is trained on a single run, it lacks the necessary historical context to identify these drops as recurrent, structured patterns within the equipment's life cycle. Lacking the exposure to multiple maintenance cycles, the network interprets these transient events as isolated anomalies or erratic fluctuations.
The resulting "jitter" and high-frequency oscillations observed in the forecasts (Figure 11) are symptomatic of a model that has over-memorized the noise of a specific run rather than learning the physics of the process. This phase confirms that for effective forecasting in chemical plants, the model must be exposed to the operational variability inherent in multiple runs. These findings provide a definitive methodological justification for the data augmentation strategy proposed in this work for the furnace dataset: without generating synthetic variations to compensate for the single-run limitation, the transition to deep learning architectures would result in unusable and unstable predictions.

6.3. Comparative evaluation and long-term robustness

In this section, the experimental results obtained by applying the PREDator framework to the two selected industrial case studies are presented and discussed. The primary objective is to validate the effectiveness of the proposed LSTM architecture, combined with a direct multi-step forecasting strategy, in providing robust predictions over extended temporal horizons (H_{long}), thereby overcoming the generalization limits observed in traditional regression models. The analysis begins with a qualitative evaluation of the model's ability to follow the non-linear profiles of key process variables in the batch environment, followed by an assessment of the forecasting performance on the continuous industrial furnace case. After that, a technical discussion on the training process is provided to ensure the absence of overfitting and the stability of the learning phase through the analysis of loss curves. Finally, the proposed LSTM framework is compared both qualitatively and quantitatively against Gaussian Process Regression (GPR), using Root of the Mean Squared Error (RMSE) as the primary metric to justify the accuracy gain achieved.

6.3.1 Qualitative Analysis: Batch Fermentation Case



(a) Batch fermentation – short horizon

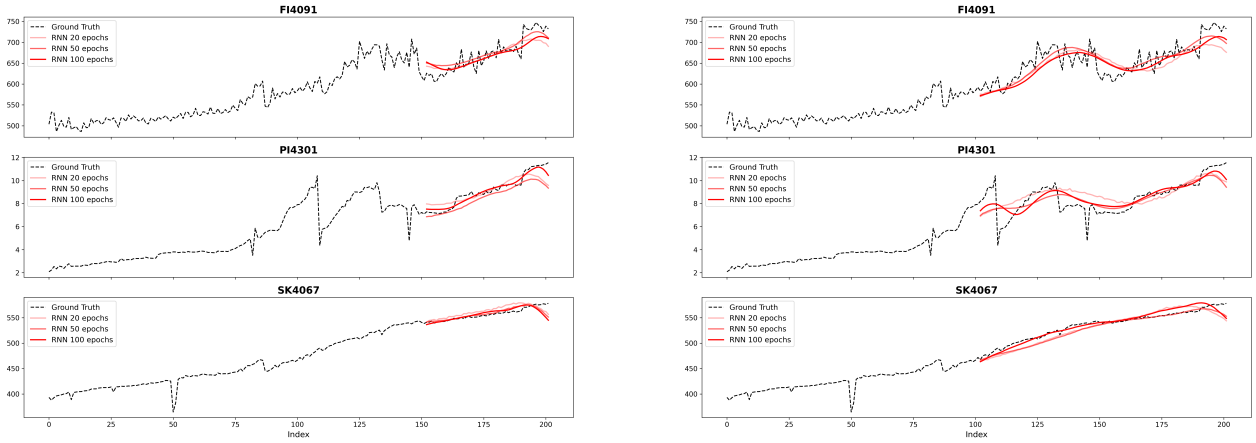
(b) Batch fermentation – long horizon

Figure 12: Qualitative comparison between Ground Truth and LSTM multi-step predictions for the batch fermentation case study. The plots illustrate the model's ability to track the non-linear profile of the Molasses Feed, the monotonic evolution of the Tank Level, and the controlled oscillations of the Temperature over a long and short-term horizons.

The application of the proposed framework to the batch fermentation case study highlights the LSTM architecture's ability to synchronize with the discrete and non-linear events characterizing the process, showing generally excellent trends across all primary variables. In the prediction of the *Molasses Feed* rate, the model accurately identifies the sequence of feeding pulses and the subsequent stabilization phases, effectively replicating the behavior without the typical smoothing artifacts of traditional regressive methods. This precision is particularly evident in the final stage of the batch, where the model correctly forecasts the complete cessation of feeding, a critical point for preventing substrate inhibition.
Simultaneously, the results for the *Tank Level* demonstrate a robust tracking of the cumulative volume increase,

maintaining a linear growth that closely matches the Ground Truth throughout the H_{long} horizon. It is observed, however, that despite the model's superior performance, increasing the number of training epochs does not lead to a massive improvement in accuracy.
Regarding the *Temperature* profile, a slight overshoot is detectable in the long-horizon predictions. This discrepancy is not a structural failure of the architecture but rather stems from the specific test run analyzed, which exhibited anomalous thermal behavior compared to the historical trends present in the training set. Despite this outlier, the model succeeds in capturing the thermal oscillations induced by the cooling system's control. Overall, the qualitative alignment confirms that the direct multi-step strategy successfully prevents the error accumulation typically observed in recursive architectures, providing a reliable digital twin of the batch evolution.

6.3.2 Qualitative Analysis: Industrial Furnace Case



(a) Industrial furnace – short horizon

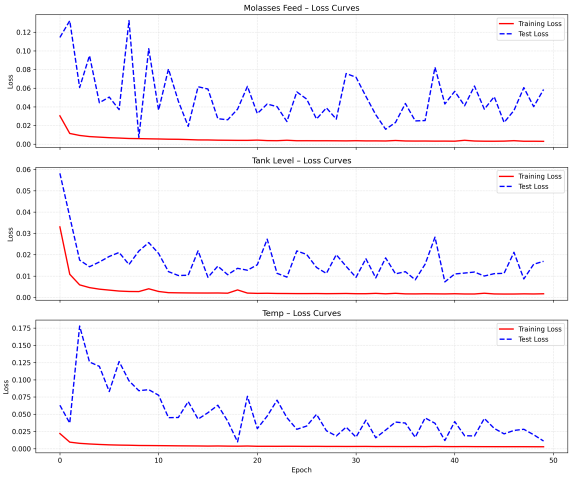
(b) Industrial furnace – long horizon

Figure 13: Performance of the LSTM model on the Industrial Furnace case study. The graph demonstrates the stability of the long-term prediction compared to the ground truth. The characteristic downward trend visible at the end of the horizon reflects the model's learned anticipation of the cycle termination (maintenance stop), consistent with the periodic nature of the training data.

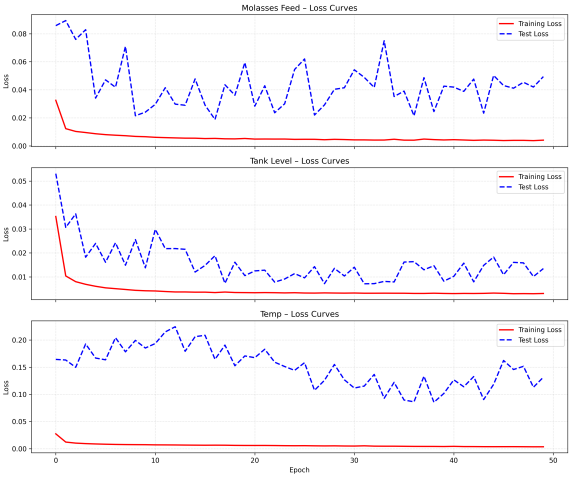
The results obtained for the industrial furnace case study highlight the superior generalization capability of the proposed LSTM framework. While an RNN trained on a single run typically suffers from overfitting, the proposed model demonstrates improved predictive performance across the validation set. The predictions exhibit remarkable solidity even over the long-term horizon (H_{long}), effectively tracking the gradual drift in process variables caused by the accumulation of deposits on the tube walls, without succumbing to the numerical instability often observed in iterative forecasting methods.
Similar to the observations in the fermentation case, it is noted that increasing the number of training epochs beyond a certain threshold does not yield an extreme improvement in the forecasting error. This suggests that the model reaches a plateau where the fundamental physical trends of the furnace are already well-captured, and further training does not significantly refine the predictions due to the inherent stochasticity of industrial sensor data.
A specific behavior observed in the final segment of the prediction horizon warrants discussion: the trajectories tend to exhibit a slight downward inflection. This phenomenon is not an artifact of the architecture but rather a consequence of the alternating nature of the furnace cycles present in the training data. Since the dataset consists of operational runs followed by maintenance stops, the network has statistically learned the expected duration of a cycle. Consequently, as the prediction extends towards the far end of the horizon, the model begins to anticipate the inevitable process shutdown, reflecting the inherent periodicity of the industrial operation. This "anticipatory" behavior confirms that the model is not merely extrapolating a linear trend but has internalized the temporal structure of the furnace's lifecycle.

6.3.3 Overall comparison

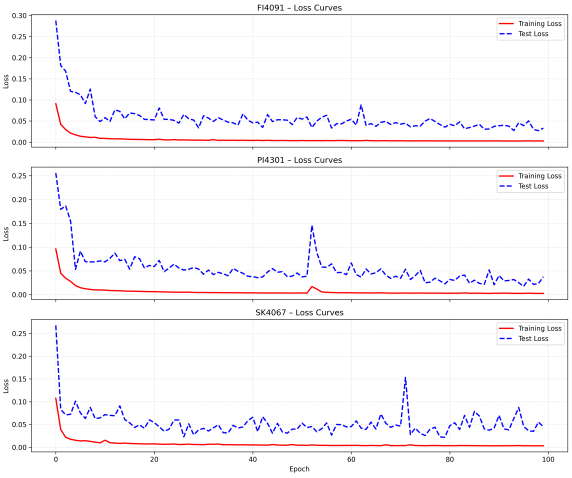
The training and test loss trajectories for the two industrial case studies are illustrated in Figure 14. In all evaluated scenarios, the training loss ($\mathcal{L}_{\text{train}}$) exhibits a sharp and consistent decay, reaching near-zero values



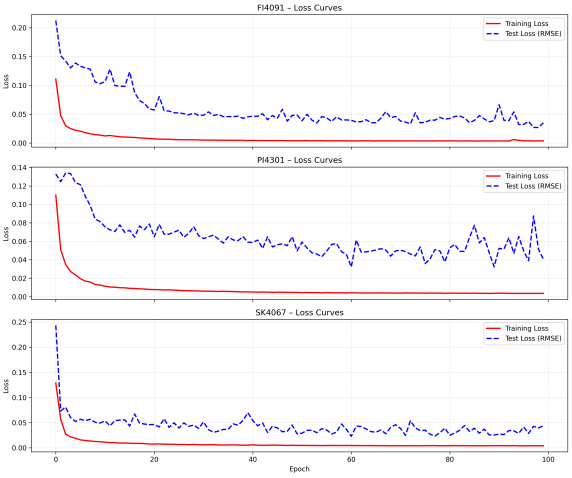
(a) Case study 1 – short horizon



(b) Case study 1 – long horizon



(c) Case study 2 – short horizon



(d) Case study 2 – long horizon

Figure 14: Training and validation loss curves of the LSTM model for the two case studies and forecast horizons.

within a few dozen epochs. This rapid convergence demonstrates that the proposed architecture possesses the requisite capacity to map the complex, non-linear dynamics of both the batch fermentation (Case Study 1) and the industrial furnace (Case Study 2).
However, a visible gap between $\mathcal{L}_{\text{train}}$ and the test loss ($\mathcal{L}_{\text{test}}$) is observed. Given the high-dimensional nature of these industrial processes and the inherent *data scarcity*, this behavior reflects the challenges of generalization under limited data availability rather than architectural limitations.

- **Case Study 1 (Batch Fermentation):** A moderate volatility is observed in the test loss (Fig. 14b,14a). The model successfully captures the global trend, although the high variance in $\mathcal{L}_{\text{test}}$ suggests that the test set contains unique batch trajectories not fully represented in the limited training pool.
- **Case Study 2 (Industrial Furnace):** The loss curves for the furnace system (Fig. 14d,14c) show higher stability, particularly in the long-horizon predictions.

Moreover, the quantitative performance of the proposed LSTM model was benchmarked against the Gaussian Process Regression (GPR) baseline using the Root Mean Square Error (RMSE) metric. As detailed in Table 4, the LSTM architecture consistently demonstrates superior forecasting accuracy across both the Batch Fermentation and Industrial Furnace case studies, particularly as the prediction horizon extends.

Industrial Furnace Case Study The performance gap between the two models is most pronounced in the Industrial Furnace dataset, specifically within the long-horizon scenarios. The GPR model exhibits significant limitations in capturing monotonic trends over extended periods. For instance, regarding the pressure variable (PI4301), the GPR model fails to retain the upward trajectory observed in the ground truth, resulting in a substantial forecast divergence (RMSE of 4.98). In contrast, the LSTM network successfully captures the underlying non-linear dynamics, maintaining the trend and reducing the error to 1.06. Similar stability is observed for variables FI4091 and SK4067, where the LSTM reduces the error by an order of magnitude compared to the GPR baseline (e.g., decreasing SK4067 error from 3.05 to 0.38).

Batch Fermentation Case Study In the Batch Fermentation process, the LSTM model proves more robust in predicting sharp structural changes. As illustrated in Figure 13, the Molasses Feed variable undergoes a rapid decline towards the end of the batch. The GPR model, biased towards smoothness, fails to anticipate this drop-off, leading to a high RMSE of 5.48. The LSTM adapts to this shift more effectively, yielding a significantly lower RMSE of 2.10. It is noted, however, that the GPR model marginally outperformed the LSTM in the long-horizon prediction of the 'Temp' variable (RMSE 0.06 vs. 0.18). This issue can be explained with the fact that while GPR is trained on one single run, the RNN has memory of the previous ones. As already discussed, the 'Temp' variable has an anomalous trend in the run used for validation: while in the cycles used for the training set a final increase towards the end is a peculiar characteristic, this does not happen in the one considered for testing. Nevertheless, the aggregate results confirm that the recurrent architecture provides a more generalized and accurate representation of the system dynamics than the probabilistic baseline.

	Batch - Short	**Batch - Long**	**Furnace - Short**	**Furnace - Long**
GPR	4.56/0.16/0.20	5.48/0.23/0.06	0.28/0.64/0.15	3.05/4.98/0.42
LSTM (100 ep)	2.63/0.10/0.02	2.10/0.09/0.18	0.23/0.27/0.11	0.38/1.06/0.15

Table 4: RMSE comparison between GPR and LSTM models for different case studies and forecast horizons. For the Batch Fermentation case, the errors are reported as Molasses Feed/Tank Level/Temp, while for the Furnace case FI4091/PI4301/SK4067.

7. Conclusions

This work addressed the problem of long-term forecasting in industrial time series, proposing a data-driven approach based on Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) architectures. Unlike many existing studies in the literature, which often rely on one-step-ahead recursive evaluation and therefore implicitly exploit future observations, the methodology adopted here explicitly focuses on genuine multi-step prediction without ground-truth injection. This design choice enables a realistic assessment of model performance under deployment-like conditions, where prediction errors naturally accumulate over extended horizons.

The main findings of this study demonstrate that the proposed architecture is capable of capturing the slow-evolving dynamics characteristic of industrial processes, preserving predictive accuracy over long forecasting windows. The results obtained across the two case studies, a batch fermentation reactor and an industrial furnace, highlight the model's ability to generalize across processes with markedly different dynamics, sampling rates, and noise characteristics. In both applications, the model achieved competitive performance in terms of error growth control and stability of predictions, outperforming baseline approaches and showcasing the practical advantages of properly designed long-term forecasting strategies.

From a scientific standpoint, this work contributes to the ongoing discussion on the limitations of conventional recursive forecasting schemes and emphasizes the importance of evaluation protocols that mirror real operational scenarios. It also provides evidence that appropriately structured RNN-LSTM architectures, combined with suitable preprocessing and horizon-definition strategies, remain highly effective for modeling long-term industrial behavior, even in the presence of strong nonlinearities and non-stationarity.

From an industrial perspective, the outcomes underscore how reliable long-term forecasts can support maintenance planning, early anomaly detection, and overall process optimization. By uncovering slow dynamics and degradation trends, the methodology aligns with the increasing adoption of predictive analytics within digitalization and Industry 4.0 frameworks.

Despite these contributions, several limitations remain. First, model performance inherently depends on data quality and availability, making sparse datasets or strongly unbalanced operating conditions challenging. Second, the approach, while effective, relies solely on historical observations and does not exploit physical knowledge that could mitigate extrapolation errors in unseen regimes. Finally, the computational cost associated with training deep architectures, particularly during hyperparameter tuning, may be non-trivial for very large industrial deployments.

8. Future Developments

Building on the results and limitations identified in this work, several research directions can be pursued to further improve the reliability and applicability of long-term predictive models in industrial contexts.

A first promising line of research concerns the enhancement of long-horizon accuracy. Techniques such as scheduled sampling, sequence-to-sequence learning, or diffusion-based forecasting models may help mitigate the accumulation of recursive errors and improve robustness under distributional shifts. Model ensembling and uncertainty quantification could also provide more informative predictions, particularly in safety-critical applications.

Another relevant direction involves the integration of data-driven forecasts with physics-based knowledge. Hybrid or physics-informed neural networks offer the possibility of combining the generalization capabilities of machine learning with the interpretability and extrapolation strengths of mechanistic models. Such integration would be especially beneficial in processes where first-principles constraints play a significant role in shaping system dynamics.

The framework developed in this work could also be extended to a broader set of industrial plants, variables, and operating scenarios. Applying the methodology to systems with different temporal resolutions, multi-input–multi-output configurations, or severe non-stationary behavior would allow testing its generality and identifying process-specific adaptations. Moreover, the consideration of exogenous variables, such as operating conditions, setpoints, or ambient factors, could further increase the predictive power and flexibility of the models.

Finally, a long-term objective is the deployment of the proposed forecasting approach within digital twin architectures. Embedding predictive models into real-time virtual replicas of industrial assets would enable continuous performance monitoring, proactive maintenance scheduling, and scenario simulation. This integration would turn long-term forecasting into a core component of next-generation decision-support systems, contributing to safer, more efficient, and more sustainable industrial operations.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. pages 106–112, 2014.

[3] Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Judith Bernett, David B Blumenthal, Dominik G Grimm, Florian Haselbeck, Roman Joeres, Olga V Kalinina, and Markus List. Guiding questions to avoid data leakage in biological machine learning applications. *Nature Methods*, 21(8):1444–1453, 2024.

[6] Hum Nath Bhandari, Binod Rimal, Nawa Raj Pokhrel, Ramchandra Rimal, Keshab R Dahal, and Rajendra KC Khatri. Predicting stock market index using lstm. *Machine Learning with Applications*, 9:100320, 2022.

[7] Sumit K. Bishnu, Sabla Y. Alnouri, and Dhabia M. Al-Mohannadi. Computational applications using data driven modeling in process systems: A review. *Digital Chemical Engineering*, 8:100111, 2023.

[8] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[9] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[10] François Chollet et al. Keras. `https://keras.io`, 2015.

[11] Paulo SG de Mattos Neto, George DC Cavalcanti, Domingos S de O. Santos Júnior, and Eraylson G Silva. Hybrid systems using residual modeling for sea surface temperature forecasting. *Scientific Reports*, 12(1):487, 2022.

[12] Norman R. Draper and Harry Smith. *Applied Regression Analysis*, volume 326. John Wiley & Sons, 1998.

[13] M. Salomé Duarte, Gilberto Martins, Pedro Oliveira, Bruno Fernandes, Eugénio C. Ferreira, M. Madalena Alves, Frederico Lopes, M. Alcina Pereira, and Paulo Novais. A review of computational modeling in wastewater treatment processes. *ACS ES&T Water*, 4(3):784–804, 2024.

[14] David Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

[15] David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. pages 1166–1174, 2013.

[16] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[17] Roger Frigola. *Bayesian Time Series Learning with Gaussian Processes*. PhD thesis, Apollo - University of Cambridge Repository, 2015.

[18] George Fuechsel. Garbage in, garbage out, 1950. Common programming adage attributed to IBM programmer George Fuechsel.

[19] Andrea Galeazzi, Francesco de Fusco, Kristiano Prifti, Francesco Gallo, Lorenz Biegler, and Flavio Manenti. Predicting the performance of an industrial furnace using gaussian process and linear regression: A comparison. *Computers & Chemical Engineering*, 181:108513, 2024.

[20] Carina L. Gargalo, Alina A. Malanca, Adem R. N. Aouichaoui, Jakob K. Huusom, and Krist V. Gernaey. Navigating industry 4.0 and 5.0: the role of hybrid modelling in (bio)chemical engineering's digital transition. *Frontiers in Chemical Engineering*, Volume 6 - 2024, 2024.

[21] Mohammad J Hamayel and Amani Yousef Owda. A novel cryptocurrency price prediction model using gru, lstm and bi-lstm machine learning algorithms. *Ai*, 2(4):477–496, 2021.

[22] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

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

[24] Mohammad Liton Hossain, SM Nasif Shams, and Saeed Mahmud Ullah. Time-series and deep learning approaches for renewable energy forecasting in dhaka: a comparative study of arima, sarima, and lstm models. *Discover Sustainability*, 6(1):775, 2025.

[25] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[26] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 3rd edition, 2021. Section 5.8: Evaluating forecast accuracy.

[27] Kouidri Ikram, Kaidameur Djilali, Dahmani Abdennasser, Raheem Al-Sabur, Benyekhlef Ahmed, and Abdel-Nasser Sharkawy. Comparative analysis of fouling resistance prediction in shell and tube heat exchangers using advanced machine learning techniques. *Res. Eng. Struct. Mater*, 10:253–270, 2023.

[28] Brian Kenji Iwana and Seiichi Uchida. Time series data augmentation for neural networks by time warping with a discriminative teacher. pages 3558–3565, 2021.

[29] A.K.S. Jardine, D. Lin, and D. Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006.

[30] F R Johnston, J E Boyland, M Meadows, and E Shale. Some properties of a simple moving average when applied to forecasting a time series. *Journal of the Operational Research Society*, 50(12):1267–1271, 1999.

[31] Jay Lee, Behrad Bagheri, and Hung-an Kao. Industrial big data analytics and cyber-physical systems for future maintenance and service innovation. *Procedia CIRP*, 16:3–8, 2014.

[32] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. *IEEE Internet of Things Journal*, 2025.

[33] Fredrik Lundh. An introduction to tkinter. *URL: www. pythonware. com/library/tkinter/introduction/index. htm*, 1999.

[34] Yuping Luo, Wenyang Wang, Yuyan Zhang, Muxin Chen, and Peng Shao. A transformer-based model for predicting and analyzing light olefin yields in methanol-to-olefins process. *Chinese Journal of Chemical Engineering*, 83:266–276, 2025.

[35] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010.

[36] Ibomoiye Domor Mienye and Theo G Swart. A comprehensive review of deep learning: Architectures, recent advances, and applications. *Information*, 15(12):755, 2024.

[37] T.G.; Obaido G.; Mienye, I.D.; Swart. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, 2024.

[38] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. *Introduction to Time Series Analysis and Forecasting*. John Wiley & Sons, 2021.

[39] Nicholas Niako, Jesus D. Melgarejo, Gladys E. Maestre, and Kristina P. Vatcheva. Effects of missing data imputation methods on univariate blood pressure time series data analysis and forecasting with arima and lstm. *BMC Medical Research Methodology*, 24(320):1–32, 2024.

[40] Oracle. Predictive maintenance and asset availability optimization, 2025.

[41] Shiyuan Pan, Xiaodan Shi, Beibei Dong, Jan Skvaril, Haoran Zhang, Yongtu Liang, and Hailong Li. Multivariate time series prediction for co2 concentration and flowrate of flue gas from biomass-fired power plants. *Fuel*, 359:130344, 2024.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] Marcus Perry. The exponentially weighted moving average. 06 2010.

[44] Fotios Petropoulos et al. Forecasting: theory and practice. *International Journal of Forecasting*, 38(3):705–871, 2022.

[45] Andri Pranolo, Faradini Setyaputri, Andien Paramarta, Alfiansyah Triono, Akhmad Fadhilla, Ade Akbari, Agung Utama, Aji Wibawa, and Wako Uriu. Enhanced multivariate time series analysis using lstm: A comparative study of min-max and z-score normalization techniques. *ILKOM Jurnal Ilmiah*, 16(2), 2024.

[46] Vijaysai Prasad, Mark Osborn, Shirley Au, K Ravi, Ch.Anvesh Reddy, Sunil Shah, Nishith Vora, and Anthony Gryscavage. Predictive heat exchanger efficiency monitoring. *Proceedings of HT2005 ASME Summer Heat Transfer Conference*, 4, 03 2009.

[47] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[48] Florian Schmidt. Generalization in generation: A closer look at exposure bias. *arXiv preprint arXiv:1910.00292*, 2019.

[49] Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

[50] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. pages 3285–3292, 2019.

[51] Xiaobao Song, Liwei Deng, Hao Wang, Yaoan Zhang, Yuxin He, and Wenming Cao. Deep learning-based time series forecasting. *Artificial Intelligence Review*, 58(1):23, 2024.

[52] Sang-Ha Sung, Chang-Sung Seo, Michael Pokojovy, and Sangjin Kim. A comparative analysis of pre-processing filters for deep learning-based equipment power efficiency classification and prediction models. *Applied Sciences*, 15(20), 2025.

[53] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[54] GE Vernova. Cloud-based asset performance management (apm), 2025.

[55] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[56] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[57] Yangyi Zhang, Sui Tang, and Guo Yu. An interpretable hybrid predictive model of covid-19 cases using autoregressive model and lstm. *Scientific reports*, 13(1):6708, 2023.

[58] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019.

[59] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.

Abstract in lingua italiana

La transizione verso l'Industria 4.0 ha reso la previsione basata sui dati un elemento fondamentale per migliorare l'affidabilità operativa e ottimizzare le strategie di manutenzione predittiva. Tuttavia, persiste una significativa dicotomia tra previsioni a breve termine ad alta accuratezza e la capacità di effettuare previsioni robuste a lungo termine, necessarie per il processo decisionale industriale. I modelli ricorsivi standard soffrono spesso di accumulo di errore e *exposure bias*, limitando la loro efficacia su orizzonti estesi. Questo lavoro affronta tali limitazioni proponendo un framework di Deep Learning basato su reti neurali ricorrenti Long Short-Term Memory (LSTM-RNN), che utilizza una strategia di previsione diretta multi-step per catturare dipendenze di lungo raggio senza richiedere l'iniezione di *ground truth*. La metodologia viene validata attraverso due distinti casi di studio industriali: un processo di fermentazione batch e un forno industriale soggetto a *fouling*. Un'analisi comparativa con Gaussian Process Regression (GPR) e la regressione polinomiale mostra che, sebbene i modelli probabilistici siano efficaci nel cogliere tendenze locali, l'architettura LSTM proposta garantisce una stabilità e una precisione superiori negli scenari di lungo periodo. Infine, la metodologia è integrata in PREDator, un'applicazione Python dotata di interfaccia grafica progettata per semplificare l'implementazione di questi avanzati strumenti predittivi sia in contesti industriali che accademici.

Parole chiave: Modelli Predittivi, Machine Learning, Manutenzione Predittiva

Acknowledgements

L'arte appresi di frangere il complesso,
Rendendo chiaro ciò che parve oscuro;
Ai Docenti la stima, ma il cuore è concesso
Alla Famiglia, il mio porto sicuro.

Uomo mi feci, e tra prove mature
E ingenuità condivise con leggerezza,
Gli amici, nelle nostre avventure
Talvolta insensate, furon scuola di fermezza.

Mi plauso, io che dall'ultimo banco
Vivevo sol di scherzi e goliardia;
Oggi mi ergo, e con orgoglio franco,
Stringo il traguardo della mia via.

E grazie a Gaia e alla sua disciplina,
Che mi armò di metodo per la sfida;
Ma la gioia permane per l'esser vicina,
E per l'amore che fu la mia guida.