TUM Data Innovation Lab
Munich Data Science Institute (MDSI)
Technical University of Munich
&
BayWa r.e

Final report of project:

ML-Driven Time Series Analysis for Predictive Maintenance of Wind Turbines

Authors	Anushka Singh, Cheng Yu, Qiqi Huang, Sicheng Dong, Jakob Bach
Mentor	M.Sc. Milena Zurmühl
TUM Mentor	M.Sc. Jona Klemenc
Project lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Jul 2025

Acknowledgements

We are grateful to Dr. Ricardo Acevedo Cabra and Prof. Dr. Massimo Fornasier for their vision and dedication in leading the TUM Data Innovation Lab, creating an environment where impactful collaborations and innovative projects can thrive. We also appreciate the continuous support of the Munich Data Science Institute (MDSI) in fostering research and providing the resources that made this work possible.
A very special thanks goes to Milena Zurmühl, our mentor from BayWa r.e., for her outstanding mentorship, unwavering support, and invaluable insights throughout the project. Her guidance was instrumental in refining our approach, navigating challenges, and ensuring the success of our research. We truly appreciate her dedication and the time she invested in our work.
We also extend our gratitude to Jona Klemenc for his valuable feedback during our monthly presentation, which helped us refine our work and improve our research direction.
Finally, we sincerely appreciate the support of BayWa r.e. and all project collaborators. We are especially grateful to BayWa r.e. for providing the necessary hardware and computational resources that enabled the successful execution of our research. This project was made possible through the collective effort and commitment of everyone involved, contributing to advancements in wind turbine failure detection.

Abstract

In the context of climate change, the generation of clean and renewable energy has become increasingly vital. This project systematically compares different modeling paradigms for temperature prediction in wind turbines. Although deployed on synthetic data, the framework demonstrates the potential to identify abnormal thermal behavior through modeling discrepancies. This contributes to improved operational efficiency, enhances system reliability, and ultimately supports the cost-effectiveness and scalability of wind energy production.

We evaluate a range of modeling approaches on a high-resolution SCADA dataset from a single wind turbine, spanning over five years and comprising more than 5 million records. These approaches include linear regression, XGBoost, a physics-informed empirical model, and deep learning architectures such as LSTM, BiLSTM, and DSCNN-BiLSTM with attention. The empirical model provides physical interpretability and performs robustly under healthy operational conditions. However, deep learning models significantly outperform traditional baselines, with the DSCNN-BiLSTM-Attention model achieving the lowest RMSE and highest R^2 score of 0.940.

This work demonstrates the potential of hybrid modeling approaches for condition monitoring in industrial settings, combining predictive accuracy, domain insight, and practical deployability in the context of renewable energy systems.

Contents

Abstract **2**

1 Introduction **4**
1.1 About BayWa r.e. . . . 4
1.2 Problem Definition and Goals of the Project . . . 4
1.3 State-of-the-Art Approaches in This Domain . . . 4

2 Data Preprocessing **5**
2.1 Data Acquisition . . . 5
2.2 Preprocessing Steps . . . 5
2.2.1 Dataset Cleaning and Splitting . . . 5
2.2.2 Missing Value Handling Process . . . 5
2.2.3 Artificial Anomaly and Trend Injection . . . 6
2.3 Feature Engineering . . . 7
2.3.1 Correlation Analysis . . . 7
2.3.2 Dimensionality Reduction using Autoencoders . . . 7
2.3.3 Domain Experts feature selection . . . 9

3 Model Selection and Building **9**
3.1 Machine Learning Methods . . . 10
3.1.1 Linear Regression . . . 10
3.1.2 XGBoost . . . 10
3.2 Empirical-Physical Modeling . . . 11
3.3 Deep Learning Methods . . . 12
3.3.1 LSTM . . . 12
3.3.2 BiLSTM . . . 13
3.3.3 DSCNN-BiLSTM . . . 13

4 Results **14**
4.1 Evaluation Metrics . . . 15
4.2 Results for Each Model . . . 16
4.2.1 Linear Regression . . . 16
4.2.2 XGBoost . . . 17
4.2.3 Empirical-Physical . . . 18
4.2.4 LSTM Family . . . 20
4.3 Results for different wind speed subsets . . . 22
4.4 Scenario-based Intervention Validation . . . 22

5 Conclusions **23**
5.1 Future Work . . . 24
5.1.1 Multi-Turbine Data Preparation . . . 24
5.1.2 Model Deployment and Monitoring . . . 24
5.1.3 Integration of Domain Knowledge . . . 24

Appendix **26**

1 Introduction

1.1 About BayWa r.e.

BayWa r.e. is a globally active renewable energy company engaged in the planning, construction, and management of wind and solar energy projects. In addition to energy trading and digital energy solutions, BayWa r.e. operates and maintains a large fleet of wind turbines across multiple geographies. A central focus of their innovation strategy is to integrate data-driven and physics-informed approaches to improve operational efficiency, reduce downtime, and enable predictive maintenance. This project was carried out in collaboration with BayWa r.e. to investigate advanced methodologies for predictive modeling based on real-world wind turbine telemetry data.

1.2 Problem Definition and Goals of the Project

Modern wind turbines are complex cyber-physical systems operating under dynamic environmental and mechanical conditions. Deviations in component temperatures—particularly in the gearbox and generator—often signal mechanical stress, lubrication failure, or incipient faults [1, 2]. If not detected early, these anomalies can lead to costly downtime and irreversible damage.
Traditional rule-based monitoring systems lack the flexibility to adapt across operational regimes or detect subtle, early-stage deviations. To address this limitation, we conduct a comparative study of three modeling paradigms—machine learning methods, empirical-physical models, and deep learning architectures. By evaluating these approaches side by side, we assess their respective strengths in leveraging domain knowledge and data-driven pattern recognition for predictive performance.
By systematically comparing these modeling paradigms, we aim to:

- Improve the accuracy and reliability of temperature forecasts.
- Demonstrate the potential of detecting abnormal trends through modeling discrepancies on artificial data.

This work supports the development of intelligent condition monitoring tools that enhance turbine reliability, reduce maintenance costs, and contribute to the broader goal of sustainable energy system resilience.

1.3 State-of-the-Art Approaches in This Domain

Recent advances in predictive maintenance for wind turbines have shifted from rule-based heuristics to data-driven and model-based strategies. Traditional statistical models, such as linear regression and ARIMA, offer interpretability and simplicity but often fall short in capturing nonlinear and multivariate dependencies present in SCADA data [3, 4].
Gradient boosting algorithms, particularly XGBoost [5], have demonstrated strong performance on structured operational datasets, with built-in feature selection and robustness to missing values [6]. Physics-informed models [7, 8] provide domain-grounded interpretability and numerical stability, particularly under healthy operating conditions, and are commonly used as baselines for evaluating data-driven alternatives.

Deep learning architectures—especially Long Short-Term Memory (LSTM) [9] and Bidirectional LSTM (BiLSTM) [10] networks—have proven effective in modeling temporal dependencies and long-range interactions in time series. More recently, hybrid deep learning models such as DSCNN-BiLSTM [11] combine convolutional and recurrent structures to better capture both spatial and temporal features in complex operational data.
This project benchmarks these diverse modeling paradigms and systematically analyzes their predictive performance, interpretability, and robustness in the context of wind turbine temperature forecasting.

2 Data Preprocessing

2.1 Data Acquisition

Our main dataset contains telemetry data from approximately 400 wind turbines, covering a historical period of 5 years. The data is collected at 10-minute intervals, providing the mean, minimum, maximum, and standard deviation for each interval.
The measured signals encompass various aspects, including measurement identification, environmental conditions, mechanical system parameters, power measurements, energy flow, electrical parameters, grid connection parameters, theoretical power output, availability, and temperature signals. This dataset offers a comprehensive foundation for research and applications in wind turbine condition monitoring, health assessment, and performance modeling.

2.2 Preprocessing Steps

To train device-based models, we first focus on data from a single device (`device_id` = 29) in the wind farm 4 (`site_id` = 4), consisting of 5,592,960 records and 87 variables.

2.2.1 Dataset Cleaning and Splitting

To analyze the impact of operational features on the target temperature, we retained only operational data by removing unrelated columns, such as `temperature_gearbox _bearing_lss` and `temperature_gear_oil_sump`. Meanwhile, we filtered the data using the `data_availability` indicator, which marks whether each telemetry record was successfully transmitted at the given timestamp. This field is essential for evaluating data completeness and communication quality. By filtering with this indicator, we exclude invalid records caused by communication failures or data acquisition errors, ensuring that our analysis and modeling are based on reliable observations.
To prevent data leakage, we split the dataset as follows: data from 2020–2022 is used for training, data from 2023 for validation, and data from 2024 and 2025 for testing. The numbers of records in each set are 154,036, 50,785, and 66,056, respectively.

2.2.2 Missing Value Handling Process

Missing values are common in wind power SCADA time series data. We first remove columns that are completely missing. For partially missing signals, we adopt a systematic approach: according to studies by MDPI [12] and Tawn [13], missingness can be categorized as either random missing (sporadic, non-systematic gaps) or continuous missing (long gaps caused by sensor failures or communication interruptions). To distinguish between short and long gaps, we set a threshold (e.g. six consecutive samples, about one hour). Short gaps are suitable for imputation, while long gaps should be removed to avoid introducing noise. For short gaps, we use the LOCF strategy, filling missing values with the most recent valid observation. In Spark, we recommend using `last()`, which handles leading and trailing gaps better than the traditional lag function. After imputation, we verify that no residual nulls remain and assess the data by checking extreme values, means, standard deviations, and visualizations. The same cleaning logic is applied to the test set to prevent data leakage and support generalization.

2.2.3 Artificial Anomaly and Trend Injection

For further model testing, we generated artificial test data based on the test set. We systematically injected various trends and anomalies into the temperature time series, including intervals with constant temperature, stepwise or continuously increasing (linear or exponential growth), as well as high-temperature spikes and periodic anomalies under specific conditions. All modifications were made by overwriting or creating new temperature signal columns. The resulting dataset contains a rich variety of anomalies and trends for subsequent model evaluation and algorithm development.The modification intervals are illustrated in Figure 1.

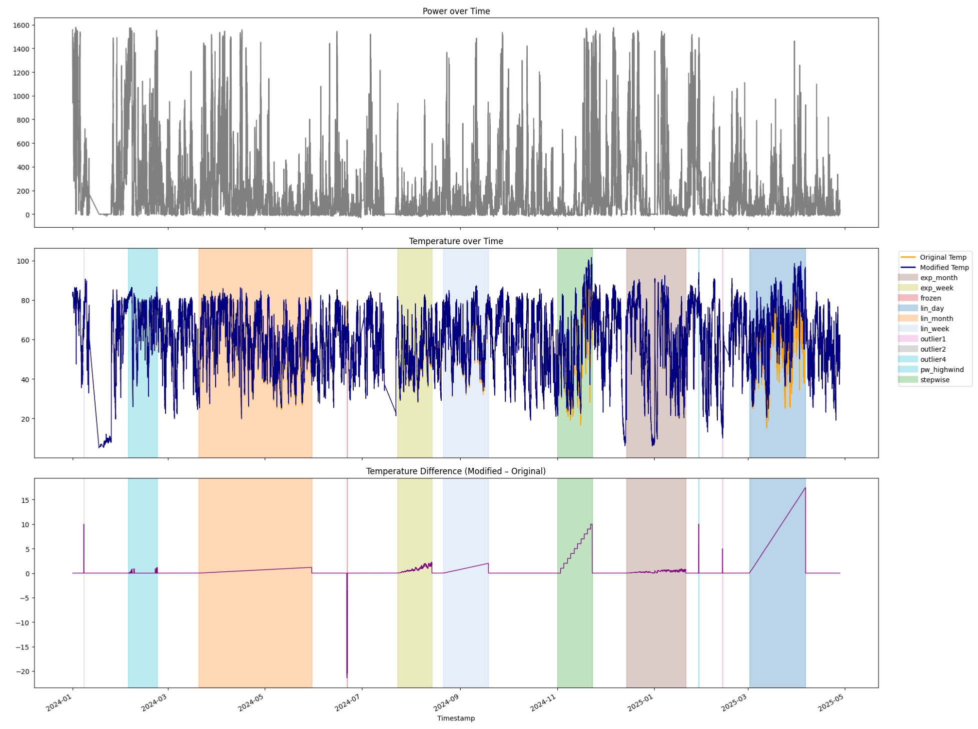

Figure 1: The artificial test set

2.3 Feature Engineering

2.3.1 Correlation Analysis

Initially, we replaced the raw operational features with their delta values ($\Delta x_t = x_t - x_{t-1}$) to examine the relationship between short-term feature changes and the target temperature. However, correlations with actual temperature or temperature deltas were negligible, so we discarded this method and did not include it in subsequent stages.
To evaluate dependencies between input features and the prediction target, both Pearson and Spearman correlation coefficients were computed:

$$\rho_{\text{pearson}}(X,Y) = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}, \quad \rho_{\text{spearman}}(X,Y) = \rho_{\text{pearson}}(\text{rank}(X), \text{rank}(Y))$$

Spearman correlation values were generally higher than Pearson values, suggesting that many features exhibit nonlinear but monotonic relationships with the target variable. The top-ranked features based on Spearman correlation are shown in Figure 2. These top 20 features were selected as one of the feature selection strategies for model training.

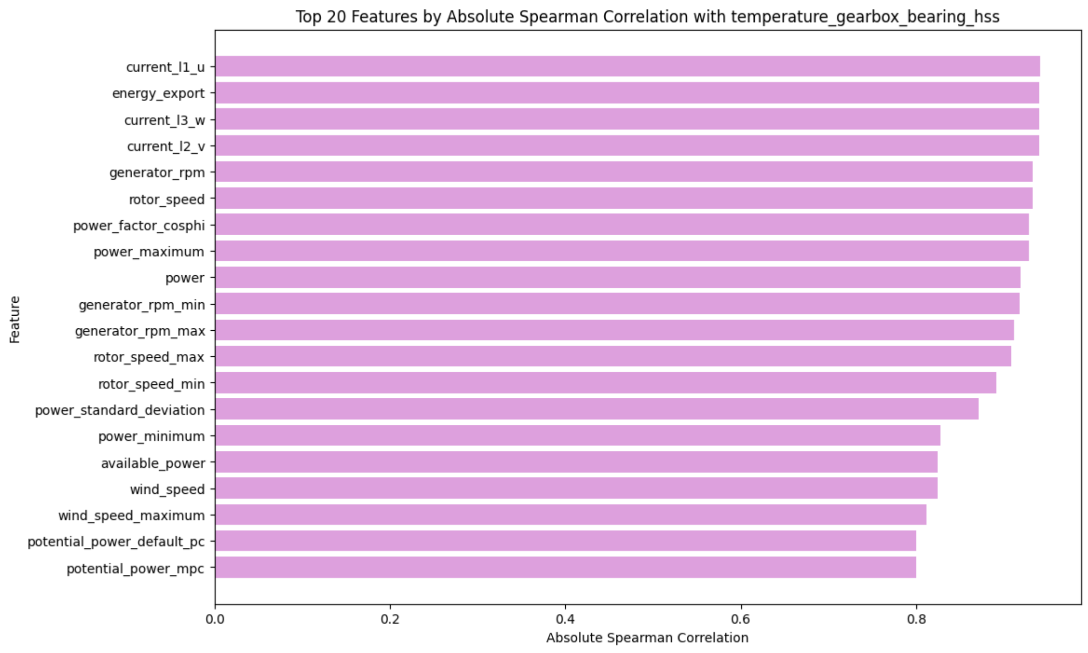

Figure 2: spearman correlation analysis results

2.3.2 Dimensionality Reduction using Autoencoders

To manage the high dimensionality and complexity of our data, we implemented an autoencoder as a data compression technique. Autoencoders are unsupervised neural networks that learn to reconstruct their inputs after mapping them to a lower-dimensional latent representation, this makes them particularly effective for dimensionality reduction in large-scale, high-frequency sensor data.

Formally, an autoencoder consists of two components: an *encoder* function $f_\theta(x)$ and a *decoder* function $g_\phi(z)$. Given an input vector $x \in \mathbb{R}^D$, the encoder maps it to a latent vector $z \in \mathbb{R}^d$, and the decoder attempts to reconstruct the original input as $\hat{x} \in \mathbb{R}^D$:

$$z = f_\theta(x), \quad \hat{x} = g_\phi(z)$$

The model is trained by minimizing the reconstruction loss between the input and its reconstruction. We used Mean Squared Error (MSE) as the loss function:

$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2 = \|x - g_\phi(f_\theta(x))\|^2$$

Our specific architecture consisted of two fully connected layers in both the encoder and decoder. The encoder compresses the input from dimension D to an intermediate hidden layer of size 64 with ReLU activations, and then to a 20-dimensional latent vector. The decoder mirrors this structure, reconstructing the original input from the latent representation. The architecture can be summarized as:

- **Encoder:** $x \in \mathbb{R}^D \to \text{Linear}(D \to 64) \to \text{ReLU} \to \text{Linear}(64 \to 20) \to z \in \mathbb{R}^{20}$
- **Decoder:** $z \in \mathbb{R}^{20} \to \text{Linear}(20 \to 64) \to \text{ReLU} \to \text{Linear}(64 \to D) \to \hat{x} \in \mathbb{R}^D$

We trained the autoencoder using the Adam optimizer (learning rate = 0.001). The autoencoder was trained on the training set only, and the encoder was later used to generate compressed representations of the validation and test sets without retraining.
The resulting 20-dimensional latent vectors served as compact inputs for our models. This significantly reduced the input dimensionality while preserving essential operational patterns, improving both training efficiency and model generalization. The close alignment between original and reconstructed signals (as shown in Figure 3) demonstrates the encoder's ability to capture key features with high accuracy.

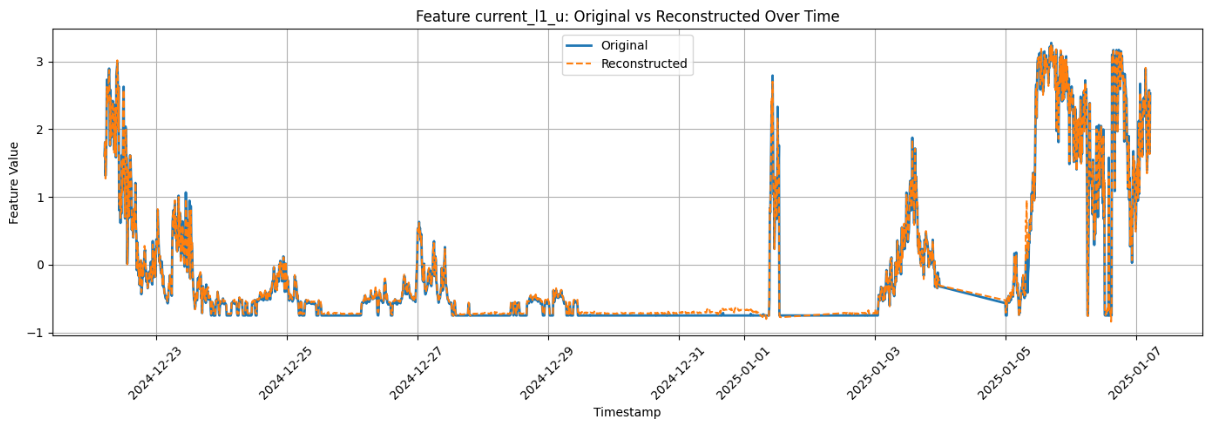

Figure 3: The original and ae-reconstructed values of a chosen feature on the test set

2.3.3 Domain Experts feature selection

Table 1: Mapping of Domain Expert-Selected Features to Dataset Columns

Descriptor Name	**Unit**	**Description**	**Matched Features in dataset**
Gen_RPM_Max	[rpm]	Maximum generator rpm	generator_rpm_max
Gen_RPM_Min	[rpm]	Minimum generator rpm	generator_rpm_min
Gen_RPM_Avg	[rpm]	Average generator rpm	generator_rpm
Gen_RPM_Std	[rpm]	Standard deviation of generator rpm	generator_rpm_standard_deviation
Rtr_RPM_Max	[rpm]	Maximum rotor rpm	rotor_speed_max
Rtr_RPM_Min	[rpm]	Minimum rotor rpm	rotor_speed_min
Rtr_RPM_Avg	[rpm]	Average rotor rpm	rotor_speed
Amb_WindSpeed_Max	[m/s]	Maximum wind speed	wind_speed_maximum
Amb_WindSpeed_Min	[m/s]	Minimum wind speed	wind_speed_minimum
Amb_WindSpeed_Avg	[m/s]	Average wind speed	wind_speed
Amb_WindSpeed_Std	[m/s]	Standard deviation of wind speed	wind_speed_standard_deviation
Amb_WindDir_Relative_Avg	[°]	Average relative wind direction	wind_direction
Amb_WindDir_Abs_Avg	[°]	Average absolute wind direction	*Not Found*
Amb_Temp_Avg	[°C]	Average ambient temperature	ambient_temperature
Prod_LatestAvg_ActPweGen0	[Wh]	Active power (generator disconnected)	energy_export
Prod_LatestAvg_TotActPwr	[Wh]	Total active power	power
Prod_LatestAvg_ReactPwrGen0	[VArh]	Reactive power (generator disconnected)	*Not Found*
Prod_LatestAvg_TotReactPwe	[VArh]	Total reactive power	reactive_power

In this study, we adopt the feature selection strategy proposed in Jankauskas's work on wind turbine gearbox temperature prediction [14], in which input variables are selected based on domain expert knowledge. As shown in Table 1, these variables are considered to have strong theoretical or empirical relevance to the target variable. We cross-referenced the list of features selected by domain experts with the available attributes in our dataset. Only the overlapping variables were retained to ensure consistency and to make the model inputs comparable.

Next Steps

We trained supervised models to predict turbine component temperatures using three different feature-processing strategies. The first approach applies Spearman rank correlation to capture strong monotonic relationships without assuming linearity. The second uses an autoencoder, selecting features based on reconstruction error to represent complex nonlinear structure. The third relies on expert-curated features to ensure interpretability. The following sections detail these models and evaluate their performance on the same test dataset. This framework allows us to assess static feature selection and the additional benefit of including temporal information.

3 Model Selection and Building

In this section, we explore and implement various modeling paradigms, ranging from traditional machine learning to deep learning and physics-informed approaches. The goal is to benchmark their performance in predicting wind turbine component temperatures.

3.1 Machine Learning Methods

3.1.1 Linear Regression

Linear Regression is a fundamental machine learning model used to estimate the relationship between input features and a continuous target variable. The model predicts an output value $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_N]^T$ as a linear combination of the weight vector w and the data matrix $X = [x_1, x_2, ..., x_N]^T$.

$$\hat{y}_i = \sum_{j=0}^{N} \phi(x_{ij}) w_j = \phi(x_i)^T w \quad therefore \quad \hat{\mathbf{y}} = \phi(X) w$$

,
Here, $\phi(\cdot)$ is a basis function that maps input features to a higher-dimensional space, allowing the model to capture nonlinear relationships such as polynomial trends.
In the context of time series data, linear regression can model dependencies by incorporating lagged values—i.e., observations from previous time steps—as additional features. This allows the model to capture temporal trends and patterns by appending lagged values to the features of the original data matrix X.
To prevent overfitting, a regularization term is often added, resulting in Ridge Regression. This approach penalizes large weight magnitudes, encouraging the model to generalize better to unseen data:

$$R = ||w||_2^2$$

The final loss function combines the mean squared error with the regularization term, weighted by a hyperparameter α

$$L(X, y, w) = ||y - \phi(X)w||_2^2 + \alpha \cdot ||w||_2^2$$

3.1.2 XGBoost

Extreme Gradient Boosting is an efficient and scalable machine learning algorithm based on the Gradient Boosting Decision Tree (GBDT) framework. It belongs to the family of ensemble learning methods, specifically the boosting category, where multiple weak learners (typically decision trees) are combined to produce a strong predictive model.
XGBoost is a supervised learning algorithm that can be applied to both regression and classification tasks. It utilizes an additive model and employs gradient descent to iteratively optimize the objective function.
Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^n$, XGBoost makes predictions by summing the outputs of K regression trees:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

where $\mathcal{F}$ denotes the space of regression trees.The overall objective function to be minimized is:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where l is a differentiable loss function, and $\Omega(f)$ is a regularization term to control model complexity.
At each iteration, a new tree is added to fit the residuals of the current prediction, and second-order Taylor expansion is used to efficiently optimize the loss.
XGBoost is well suited for this task as it efficiently handles structured tabular data and provides automatic feature importance evaluation, which assists in understanding how operational parameters influence the target temperature [15]. Also, its built-in regularization (L1/L2) helps prevent overfitting, ensuring robustness and generalization when working with large-scale industrial datasets [5]. Moreover, XGBoost natively supports handling missing values and outliers by learning optimal split directions, reducing the need for extensive preprocessing. Finally, its computational efficiency during training and prediction makes it a practical choice for condition monitoring and predictive maintenance of wind turbines.

3.2 Empirical-Physical Modeling

We employ a physics-informed empirical modeling approach grounded in a reduced-order thermal system representation, following the methodology established in prior work [8]. This approach provides robust interpretability through the direct correspondence between observable variables and underlying physical processes governing heat generation and transfer. The fundamental model structure consists of a first-order linear ordinary differential equation (ODE):

$$C \cdot \dot{T}(t) = C \cdot \frac{dT(t)}{dt} = \dot{Q}_{+}(t) + hA \cdot (T_{\text{ext}}(t) - T(t))$$

where C is the heat capacitance, $T(t)$ the component temperature, $T_{\text{ext}}(t)$ the ambient (sink) temperature, and hA the convective heat transfer coefficient.
The heat input $\dot{Q}_{+}(t)$ is empirically modeled as:

$$f(t) = f_0 + k \cdot T_{\text{ext}}(t) + \sum_{i=1}^{n} \left(a_i \cdot P^i(t) + b_i \cdot \omega^i(t) \right)$$

where f_0, k, a_i, and b_i are regression parameters fitted from operational data to capture nonlinear effects.
To numerically simulate the system dynamics, the Forward-Euler integration method is adopted:

$$\Delta T = \frac{\Delta t}{\tau} \left(f(t) - k \cdot T(t) \right)$$

Here, τ is a modeling parameter introduced to represent the system's thermal time constant. While not derived from first-principles physics, τ effectively captures the thermal inertia and contributes to numerical stability by smoothing abrupt variations during integration. This scheme offers a favorable balance between physical interpretability, predictive accuracy, and computational efficiency in modeling the thermal behavior of wind turbine components.

Table 2: Parameter options evaluated in the analytical model grid search.

Parameter	**Options**
Target temperature	`temperature_gearbox_bearing_hss`
Heat sink proxies	`nacelle_temperature`, `ambient_temperature`
Rotational speed	`rotor_speed`, `generator_rpm`
Power input	`power`
Polynomial order	0,1,2,3,4

Experimental Configuration Model configurations were evaluated as follows:
The top two performing configurations both targeted `temperature_gearbox_bearing_hss`, using `nacelle_temperature` as the heat sink. One configuration employed `rotor_speed` and the other used `generator_rpm` as the rotational speed input. Both achieved their best performance with a polynomial order of 4.
Across all trials, predictive accuracy improved consistently with increasing order up to 4, beyond which marginal gains were negligible. As a result, we fixed the polynomial order at 4 in all subsequent modeling experiments.

3.3 Deep Learning Methods

3.3.1 LSTM

The *Long Short-Term Memory* (LSTM) network [9] is a specialized architecture within the family of Recurrent Neural Networks (RNNs), designed to address some of the limitations of standard RNNs.
Vanilla RNNs typically suffer from the *vanishing gradient problem*, where gradients become exponentially small as they are backpropagated through time. This makes it difficult for the network to learn long-range dependencies, as earlier timesteps have diminishing influence on weight updates.

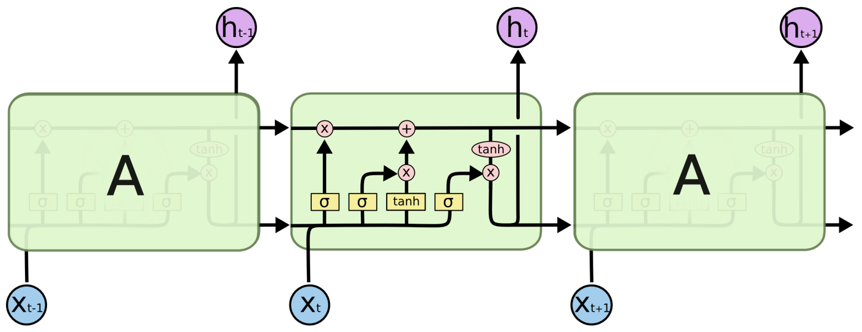

Figure 4: The structure of a LSTM predicting the future value h_{t+1} from the input sequence X_{t-1}, X_t and X_{t+1}
(Retrieved from: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

LSTMs mitigate this issue through the use of memory cells and gating mechanisms—specifically, the input, output, and forget gates. The forget gate allows the model to selectively retain or discard information, effectively controlling the flow of gradients and

preserving relevant context over longer sequences. By doing so, LSTMs are able to maintain stable gradient propagation, making them capable of learning both short- and long-term dependencies.
Thanks to these architectural innovations, LSTMs are particularly effective for sequence modeling tasks such as time series forecasting, anomaly detection, natural language processing, and any domain where the temporal structure of data is critical.
This makes them able to catch long-term dependencies which are common in time-series data. For training LSTMs require data in a sliding window form, i.e. the data is divided into groups of n consecutive timestamps where the samples 1 till $n-1$ are used to predict the nth sample.

3.3.2 BiLSTM

Bidirectional LSTMs (BiLSTMs) are an extension of the standard Long Short-Term Memory (LSTM) networks. As the name suggests, BiLSTMs process input sequences in both forward (left-to-right) and backward (right-to-left) directions. This bidirectional processing allows the model to capture context from both past and future time steps, which can lead to improved performance in tasks where full sequence information is valuable.
In a study by Siami-Namini et al. [10], the authors compared standard LSTMs with BiLSTMs for time series forecasting, specifically on stock market prediction tasks. Their results showed that BiLSTMs outperformed standard LSTMs, achieving a 38% improvement in RMSE. This highlights the potential of bidirectional architectures in capturing complex temporal dependencies more effectively.
Motivated by these findings, we also chose to apply BiLSTMs to our time series data to explore whether similar performance gains could be achieved.

3.3.3 DSCNN-BiLSTM

Although the BiLSTM model has demonstrated strong performance in temperature prediction tasks, its limitations become apparent when dealing with complex, nonlinear temperature data. A single model often struggles to fully and accurately capture all underlying trends. This has led to the increasing use of hybrid models in the literature.
In this chapter, we propose an enhanced version of the BiLSTM model by integrating a **Depthwise Separable Convolutional Neural Network (DSCNN)** layer to form a hybrid model, inspired by the paper from Xinping Li. [11] In the original study, the input data consisted solely of historical temperature sequences. In this work, we extend the model by incorporating various combinations of past and present operational data as input features for prediction. The DSCNN component significantly reduces the number of parameters and computational complexity while retaining strong spatial feature extraction capabilities. In parallel, the BiLSTM component captures bidirectional dependencies in the time series, thus enhancing the model's ability to learn periodic patterns and long-term dependencies in high-dimensional temperature data.

Overall architecture The detailed model architecture is illustrated in Figure 5.

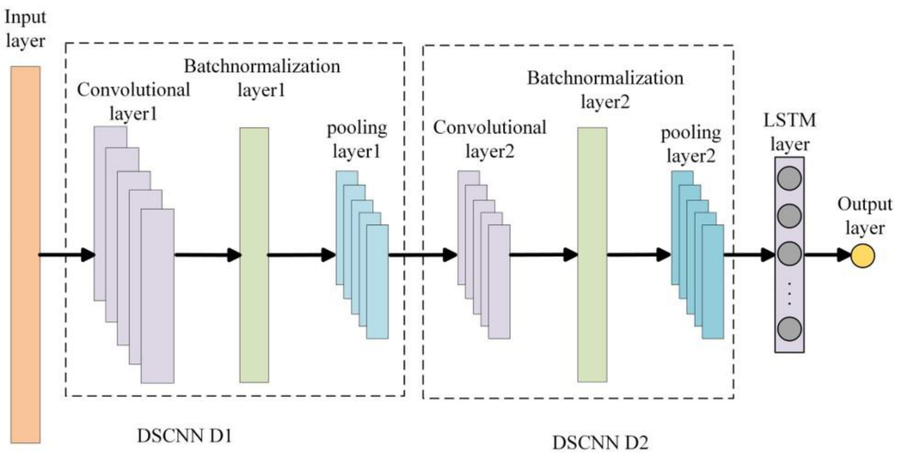

Figure 5: The overall structure of DSCNN-BiLSTM Model.

The model is composed of two sequential **Depthwise Separable Convolutional Neural Network (DSCNN)** blocks, followed by an LSTM layer and a fully connected output layer. Each DSCNN block consists of a depthwise separable convolutional layer, a batch normalization layer, and a max-pooling layer. This design allows for efficient feature extraction and dimensionality reduction before temporal modeling. The exact hyperparameter are also the same as those in the paper [11] in Table 12 (Appendix).
To further enhance the model's capacity for capturing temporal dependencies and selectively focusing on informative features, we extended the original DSCNN-BiLSTM architecture by integrating an attention layer after the BiLSTM module. This modification enables the model to assign dynamic weights to different time steps, allowing it to concentrate on more relevant parts of the input sequence when making predictions.

4 Results

Our predictive target is the temperature of the gearbox bearing high-speed shaft(reason reference). We evaluate different feature selection or data-processing techniques across multiple models.
In this study, we compare the performance of XGBoost, LSTM-based, and Empirical-Physical models for predicting the temperature of the gearbox high-speed shaft in wind turbines. The rationale for selecting these models is threefold: XGBoost is well-suited for structured tabular data and provides automatic feature importance; LSTM excels at modeling long- and short-term dependencies in time series; DSCNN–BiLSTM combines convolutional layers with bidirectional LSTM to capture both local and temporal features; empirical-physical models incorporate domain knowledge and offer physically interpretable baselines. Prior research has demonstrated that integrating tree-based, deep learning, and physics-based approaches can enhance performance and interpretability in predictive maintenance tasks [11, 8].
All models are trained, validated, and tested using the same datasets, and their performance is measured using four evaluation metrics: RMSE, R^2, MAE, and MedAE. In this chapter, we will first explain the meaning of each metric, then compare the best results for each model, and present the detailed performance of all models. Finally, we focus on the XGBoost and DSCNN–BiLSTM models and validate their performance on a manually curated test dataset.

4.1 Evaluation Metrics

RMSE
The RMSE (Root Mean Squared Error) describes the square of the average difference of each data point.
It is defined as

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N}(\hat{y}_i - y_i)^2}{N}}$$

Where N corresponds to the number of sampled. y_i and $\hat{y}_i$ describe the ground truth as well as the predicted values of the targeted feature respectively.
The smaller the RMSE value is, the closer the predictions are in relation to the ground truth.

R^2 Score
Another metric which we use to evaluate the performance of our models is the R^2 Score. It is also called *coefficient of determination* and is defined as

$$R^2 = 1 - \frac{R_{SS}}{T_{SS}}$$

Where R_{SS} and T_{SS} are the sum of squares of the residuals and the total value, respectively. They are defined as

$$R_{SS} = \sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \quad T_{SS} = \sum_{i=0}^{N}(y_i - \bar{y})^2 \quad with \quad \bar{y} = \frac{1}{N}\sum_{i=0}^{N} y_i$$

Therefore, in an optimal prediction the sum of squares of residuals would be zero, leading to a perfect R^2 score of 1.

Mean and Median Absolute Error
The *Mean Absolute Error* (MAE) and *Median Absolute Error* (MedAE) are common regression metrics based on absolute residuals.

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

MAE measures the average magnitude of errors. It is straightforward to interpret but sensitive to outliers.

$$\text{MedAE} = \text{median}\left(|y_1 - \hat{y}_1|, \ldots, |y_N - \hat{y}_N|\right)$$

MedAE captures the typical absolute error and is more robust to outliers than MAE.
Both metrics are in the same units as the target variable; lower values indicate better predictive performance.

4.2 Results for Each Model

In the following, we will compare the results of different models To make the data comparable we confined ourselves to the data of a specific turbine within a certain timespan with a presplit test and training set. This makes the model performance more comparable. Table 3 summarizes the results by comparing the aforementioned evaluation metrics. Note here that only the best models of each respective model class are mentioned in this table, in the following we describe our results in detail.

Table 3: The best result of each model class is shown in this table. In-depth results are described in the following.

Model	**RMSE**	**R^2**	**MAE**	**MedAE**
Linear Regression	9.542	0.682	5.763	5.693
XGBoost	8.407	0.785	6.001	4.493
XGBoost with lags	5.014	0.882	3.934	3.257
Emp-Physical *(Healthy)*	6.784	0.784	5.050	3.834
Emp-Physical *(Total)*	35.571	-3.212	19.476	14.661
LSTM	4.6638	0.9276	3.6152	2.8834
BiLSTM	4.3135	0.9380	3.2806	2.5617
DSCNN-BiLSTM	4.866	0.921	3.760	3.000
DSCNN-BiLSTM-Attention	4.250	**0.940**	3.218	2.499

4.2.1 Linear Regression

Table 4 presents the results of the linear regression model evaluated with varying values of regularization strength and lag features. The best R^2 score achieved was 0.682, which indicates a suboptimal fit. Furthermore the regularization strength α did not have a high impact on the performance measures.

Table 4: Prediction results with the Linear Regression Model

Features	**Regularization Strength α**	**Lag(s)**	**RMSE**	**R^2**	**MedAE**
spearman	0	None	9.577	0.680	5.337
spearman	0	[10]	9.706	0.671	5.776
spearman	**0**	**[5, 10]**	**9.542**	**0.682**	**5.693**
spearman	1	None	9.818	0.663	5.631
spearman	1	[10]	9.706	0.671	5.779
spearman	1	[5, 10]	9.600	0.678	5.781

The relatively low scores suggests that the underlying data cannot be effectively modeled using a linear approach. In particular, the model struggles to capture complex temporal patterns and long-term dependencies inherent in the time series.
Consequently, we chose not to explore linear regression further in this report and instead focused on more expressive models, such as deep learning architectures like LSTMs.

4.2.2 XGBoost

In this study, we implemented four XGBoost variants to evaluate different feature selection methods and the effect of temporal lag. The first uses Spearman rank correlation to capture strong monotonic relationships without assuming linearity. The second employs autoencoder-derived features, selected via reconstruction error to represent nonlinear structure. The third relies on expert-curated features for interpretability. The fourth builds on the autoencoder approach by adding a 10-step sliding-window of lagged variables, converting static features into a time-aware representation that encodes temporal dependencies. This systematic framework enabled us to compare static feature selection performance and the added value of incorporating temporal information.

Table 5: Performance of XGBoost models with different feature selection strategies.

Model	**Feature**	**RMSE**	**R^2**	**MAE**	**MedAE**
XGboost	autoencoder	**8.0469**	**0.7848**	**6.0008**	4.4928
XGboost	experts	8.4057	0.7652	6.0436	**4.3373**
XGboost	spearman	8.6423	0.7517	6.1744	4.3838

Among the three XGBoost variants, the autoencoder-based model performs best, achieving an R^2 of 0.7848. This indicates that autoencoder-derived features capture complex nonlinear patterns more effectively, enhancing predictive accuracy. It shows that automatic feature learning can uncover deeper patterns than traditional methods, improving generalization and practical efficiency. Nonetheless, the overall performance gap remains small, as all models extract most key information. Details are shown in Table 5.

Table 6: Performance comparison of XGBoost models with and without lagged features.

Model	**Feature**	**RMSE**	**R^2**	**MAE**	**MedAE**
XGboost	experts	8.4057	0.7652	6.0436	4.3373
XGboost lags10	experts	**5.4244**	**0.8650**	**4.2308**	**3.5556**
XGboost	autoencoder	8.0469	0.7848	6.0008	4.4928
XGboost lags10	autoencoder	**5.0142**	**0.8821**	**3.9342**	**3.2570**

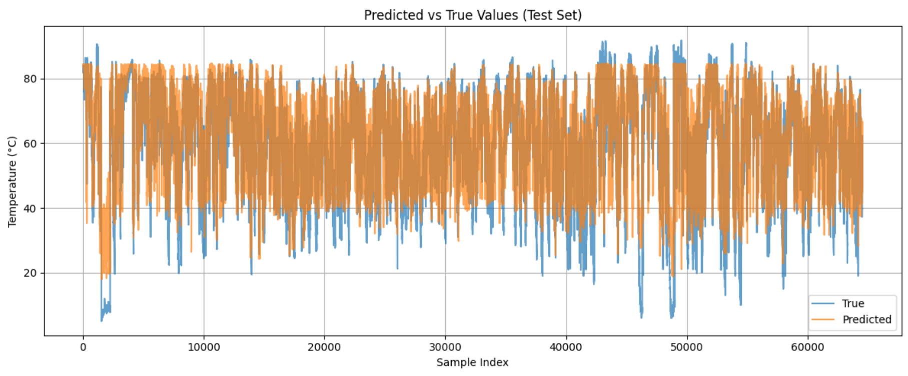

Figure 6: Prediction results for testing set by XGBoost using autoencoder

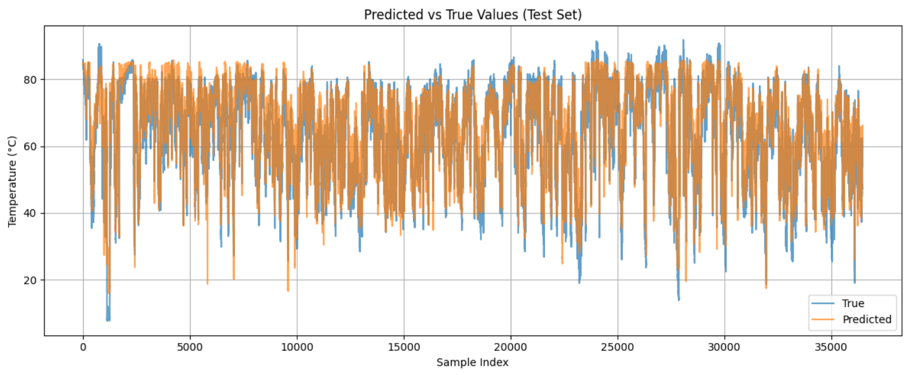

Figure 7: Prediction results for testing set by XGBoost with lags10 using autoencoder

In this experiment, we extended both expert-selected and autoencoder-derived feature sets with a 10-step lag window to enhance temporal awareness and capture autocorrelation in the target variable. Across both models, RMSE, MAE and MedAE decreased noticeably while R^2 increased substantially. This aligns with widely observed results in time-series forecasting: incorporating lagged values of the features notably improves XGBoost's predictive accuracy. Detailed outcomes are given in Table 6, with Figure 6 and Figure 7 illustrating performance before and after lag feature inclusion.

Table 11(Appendix) illustrates how the same XGBoost model, using autoencoder-derived features, performs on the original versus a synthetic test set. Its $R^2 = 0.7848$ drops to –1.280, indicating robustness of the model.

4.2.3 Empirical-Physical

Impact of Data Quality: Healthy vs. Total The empirical-physical model estimates temperature based on initial conditions, external temperature, power, and rotational speed over continuous time intervals. As it is sensitive to missing or anomalous inputs, we first identified continuous intervals marked as `valid=1`, as shown in Table 7. From this, we extracted a continuous high-quality interval (**healthy**: 2024-01-02 to 2024-01-11). For comparison, the model was also evaluated on all valid periods (**total**).

Table 7: Example of filtered time segments with validity flag

valid	start_ts	end_ts	duration
1	2024-01-01 00:00:00	2024-01-02 09:40:00	203
0	2024-01-02 09:50:00	2024-01-02 09:50:00	1
1	2024-01-02 10:00:00	2024-01-11 13:40:00	1319
0	2024-01-11 13:50:00	2024-01-17 14:20:00	868
...	...	...	...

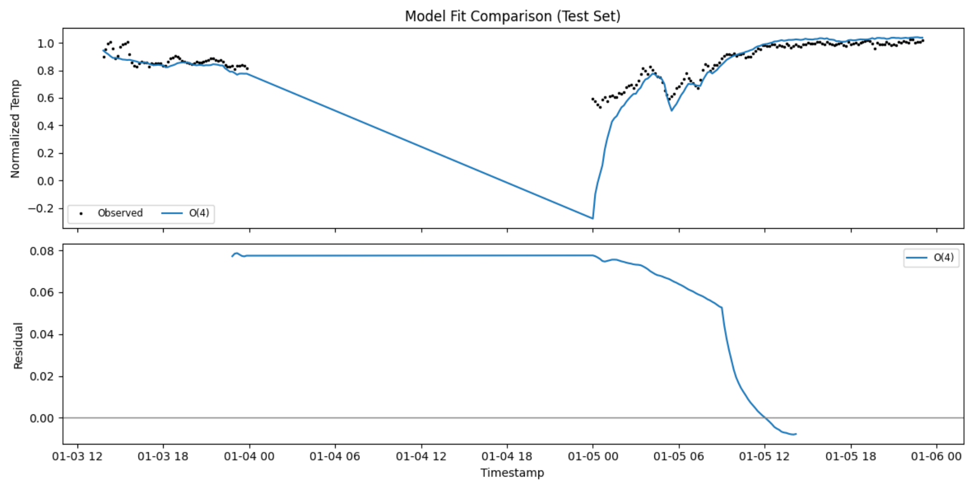

Figure 8: Impact of missing data on model fit.

As shown in Figure 8, missing data leads to poor interpolation and large post-gap residuals. The model misaligns with system dynamics, reducing reliability. Comparing performance on the healthy and total subsets quantifies this sensitivity and emphasizes the value of clean data.

Model Complexity: Effect of Polynomial Order Table 8 summarizes the performance of the Empirical-Physical model under different polynomial orders of ω (generator_rpm and rotor_speed).
Model accuracy improves notably from order 0 to 3, with R^2_{te} increasing from roughly -0.86 to 0.78 and RMSE dropping by over 10 units. The most substantial gain occurs from order 0 to 1. However, the improvement from order 3 to 4 is minimal (e.g., R^2_{te} increases from 0.782 to 0.784), indicating diminishing returns. In practice, using order 3 or 4 achieves a good balance between accuracy and model complexity.

Table 8: Empirical-Physical model performance (fixed temperature_gearbox_bearing_hss → nacelle_temperature). Abbreviations: Order = polynomial order; tr/te = train/test.

ω	Order	R^2_{tr}	RMSE_tr	MedAE_tr	R^2_{te}	RMSE_te	MedAE_te
generator_rpm	4	0.828	7.525	4.411	0.784	6.782	3.833
generator_rpm	3	0.826	7.579	4.422	0.782	6.819	3.211
rotor_speed	4	0.829	7.518	4.442	0.782	6.824	3.868
rotor_speed	3	0.826	7.580	4.422	0.781	6.832	3.210
rotor_speed	2	0.809	7.935	4.479	0.716	7.787	4.016
generator_rpm	2	0.809	7.934	4.481	0.714	7.810	4.054
rotor_speed	1	0.767	8.772	5.031	0.592	9.326	5.321
generator_rpm	1	0.764	8.820	5.080	0.580	9.467	5.472
rotor_speed	0	-0.404	21.527	17.780	-0.864	19.936	19.808
...	...	...	...	...	...	...	...

4.2.4 LSTM Family

Table 9 presents a comprehensive comparison of several LSTM models under various configurations. Several key trends emerge from the results:

BiLSTM achieves better performance compared to plain LSTM. In most configurations, the BiLSTM improves model performance, especially regarding R2. It also supports prior findings in the literature [10].

Model complexity also has obvious impact on the observation. More sophisticated architectures such as DSCNN_BiLSTM_Attention generally outperform simpler models across most evaluation metrics. However, the improvement depends heavily on the compatibility between the model architecture and the feature selection method. Autoencoder-based features provide the most robust and consistent performance among the three feature selection methods. In contrast, while offering reasonable baseline performance, domain-based features tend to yield less consistent results, especially in attention-based models. Spearman correlation-based features perform well in specific configurations (e.g., BiLSTM with large window and batch size), but are generally unstable in deeper architectures, sometimes leading to extreme outliers (e.g., MSE > 75).

Window Size plays a critical role in model performance. For relatively simple architectures such as LSTM and BiLSTM, increasing the window size from 10 to 20 does not yield performance gains and may, in some cases, lead to degradation—potentially. In contrast, more complex architectures like DSCNN_BiLSTM_Attention tend to benefit from longer input sequences generally, though some outliers can still be observed, the highest observed R^2 score (0.9398) was achieved with a window size of 20 in combination with autoencoder-derived features.

Batch Size has a notable influence. A batch size of 128 generally provides a good balance for shallow models. However, larger batch sizes can introduce instability when used with weaker features such as those based on Spearman correlation, suggesting that batch size should be carefully tuned with respect to both model depth and feature type.

In summary, the experimental results confirm several key insights. First, model complexity contributes positively to performance, but its benefit is contingent on the quality of the input features. Autoencoder-based features offer the highest overall adaptability and are particularly effective when used with deep, attention-based models. While Spearman and domain-based features can perform adequately under specific configurations, they generally lack robustness across architectures.

Table 9: Performance of LSTM, BiLSTM, DSCNN-BiLSTM and DSCNN-BiLSTM-Attention Model under different conditions

Feature Selection Method	Model Type	Window	Batch	MSE	RMSE	MAE	MedAE	R^2
domain	lstm	10	128	46.8631	6.8457	5.2888	4.4456	0.8440
	bilstm	10	128	46.4933	6.8186	5.5803	4.9408	0.8452
	dscnn_bilstm	10	128	30.2608	5.5010	4.1524	3.2320	0.8992
	dscnn_bilstm_attention	10	128	28.5499	5.3432	3.9389	2.9428	**0.9049**
domain	lstm	10	256	43.3165	6.5815	5.2453	4.4956	0.8558
	bilstm	10	256	41.0292	6.4054	5.0652	4.3004	0.8634
	dscnn_bilstm	10	256	29.9366	5.4714	4.0626	3.0433	**0.9003**
	dscnn_bilstm_attention	10	256	33.6480	5.8007	4.4034	3.4011	0.8880
domain	lstm	20	64	35.0755	5.9225	4.5664	3.6917	0.8831
	bilstm	20	64	46.5697	6.8242	5.5109	4.7850	0.8447
	dscnn_bilstm	20	64	27.6979	5.2629	4.0874	3.2981	**0.9077**
	dscnn_bilstm_attention	20	64	29.0443	5.3893	4.2662	3.5765	0.9032
domain	lstm	20	128	37.9601	6.1612	4.9810	4.3367	0.8734
	bilstm	20	128	33.9604	5.8276	4.4679	3.5695	0.8868
	dscnn_bilstm	20	128	32.4723	5.6984	4.4593	3.6839	0.8917
	dscnn_bilstm_attention	20	128	22.6985	4.7643	3.5934	2.8187	**0.9243**
domain	lstm	20	256	41.7540	6.4617	5.2639	4.6392	0.8608
	bilstm	20	256	41.9241	6.4749	5.2665	4.5981	0.8602
	dscnn_bilstm	20	256	26.9407	5.1904	3.8305	2.8775	**0.9102**
	dscnn_bilstm_attention	20	256	28.5172	5.3401	4.0834	3.2241	0.9049
spearman	lstm	10	128	44.3302	6.6581	5.2600	4.5191	0.8524
	bilstm	10	128	46.4765	6.8174	5.6139	5.0051	0.8453
	dscnn_bilstm	10	128	34.1637	5.8450	4.3072	3.3557	0.8862
	dscnn_bilstm_attention	10	128	28.8699	5.3731	3.8276	2.7802	**0.9039**
spearman	lstm	10	256	43.3672	6.5854	5.0620	4.2007	0.8556
	bilstm	10	256	48.1710	6.9405	5.5218	4.7700	0.8396
	dscnn_bilstm	10	256	34.8278	5.9015	4.4227	3.5435	0.8840
	dscnn_bilstm_attention	10	256	31.2525	5.5904	3.9585	2.9697	**0.8959**
spearman	lstm	20	128	41.2750	6.4246	5.2404	4.7040	0.8624
	bilstm	20	128	41.2759	6.4246	5.2465	4.7389	0.8624
	dscnn_bilstm	20	128	27.7215	5.2651	3.8459	2.9026	**0.9076**
	dscnn_bilstm_attention	20	128	30.6733	5.5384	4.1601	3.2731	0.8977
spearman	lstm	20	256	40.8959	6.3950	5.1908	4.6238	0.8637
	bilstm	20	256	35.2753	5.9393	4.7510	4.1518	0.8824
	dscnn_bilstm	20	256	31.9399	5.6515	4.3137	3.5503	0.8935
	dscnn_bilstm_attention	20	256	24.3491	4.9345	3.4714	2.5398	**0.9188**
autoencoder	lstm	10	128	29.2981	5.4128	4.3218	3.6841	0.9024
	bilstm	10	128	30.3312	5.5074	4.3383	3.8120	0.8990
	dscnn_bilstm	10	128	45.9006	6.7750	5.1201	3.9551	0.8472
	dscnn_bilstm_attention	10	128	24.9320	4.9932	3.9058	3.1650	**0.9170**
autoencoder	lstm	10	256	21.7509	4.6638	3.6152	2.8834	0.9276
	bilstm	10	256	18.6059	4.3135	3.2806	2.5617	**0.9380**
	dscnn_bilstm	10	256	41.1618	6.4157	4.8142	3.7265	0.8629
	dscnn_bilstm_attention	10	256	27.1009	5.2059	3.9302	3.1070	0.9098
autoencoder	lstm	20	64	33.8614	5.8191	4.0255	3.1907	0.8871
	bilstm	20	64	23.9705	4.8960	3.7750	3.1602	0.9201
	dscnn_bilstm	20	64	23.6793	4.8661	3.7601	2.9999	0.9211
	dscnn_bilstm_attention	20	64	18.0617	4.2499	3.2179	2.4986	$\boxed{\mathbf{0.9398}}$
autoencoder	lstm	20	128	29.0480	5.3896	4.3524	3.8293	0.9032'
	bilstm	20	128	27.9985	5.2914	4.0355	3.2492	**0.9067**
	dscnn_bilstm	20	128	36.7077	6.0587	4.7477	3.8951	0.8776
	dscnn_bilstm_attention	20	128	31.8762	5.6459	4.2848	3.3574	0.8937
autoencoder	lstm	20	256	32.5618	5.7063	4.6322	4.0893	0.8914
	bilstm	20	256	40.4010	6.3562	5.3022	4.8146	0.8653
	dscnn_bilstm	20	256	36.6161	6.0511	4.4933	3.3459	0.8779
	dscnn_bilstm_attention	20	256	22.5264	4.7462	3.6270	2.8843	**0.9249**

4.3 Results for different wind speed subsets

We further investigate the performance of our prediction models under different wind turbine operational states. The operational states are categorized based on wind speed: **low** (< 4 m/s), **medium** (4–13 m/s), and **high** (> 13 m/s). We selected representative models and summarized their performance metrics under each wind regime, as shown in Table 10.

When employing domain-expert feature selection, both DSCNN-BiLSTM-Attention and XGBoost models achieve their best predictive accuracy on medium wind speeds. However, under the autoencoder feature selection, both models perform poorly across different wind speed regimes.

Table 10: Performance comparison of models under different windspeed levels

Model	Feature Selection	Windspeed	RMSE	R^2	MAE	MedAE
	experts	Low	8.3246	0.5132	6.6737	5.5792
DSCNN-BiLSTM-Attention	experts	**Medium**	**4.7971**	**0.8553**	**3.6026**	**2.9135**
	experts	High	15.7048	0.7677	14.3839	11.9403
	autoencoder	Low	20.5182	-1.9571	17.1665	15.5979
DSCNN-BiLSTM-Attention	autoencoder	Medium	12.9806	-0.0595	8.1430	5.2526
	autoencoder	**High**	**30.1337**	**0.1448**	**26.4102**	**19.8749**
	experts	Low	9.6103	0.3579	7.6433	6.5129
XGBoost	experts	**Medium**	**7.0150**	**0.6945**	**4.7983**	3.3195
	experts	High	14.2430	0.7758	7.9414	**2.0791**
	autoencoder	Low	19.8399	-1.7365	16.7205	15.6021
XGBoost	autoencoder	**Medium**	**13.3820**	**-0.1119**	**9.0337**	**6.7918**
	autoencoder	High	32.4921	-0.1666	23.5458	12.7409

4.4 Scenario-based Intervention Validation

To assess the sensitivity of the models to variations in key input features, we employed a *scenario-based intervention validation* approach. Specifically, we manually modified the remote sensing data for the years 2024 and 2025 to introduce hypothetical trends, as illustrated in Figure 1. This technique allows us to evaluate whether the models are capable of capturing anticipated patterns that are not present in the original dataset. We applied this modified dataset to three different models: a Deep Separable Convolutional Neural Network (DSCNN), XGBoost, and a traditional statistical model. By comparing their predictions against the manually embedded trends, we aim to examine each model's ability to respond appropriately to the imposed data shifts and assess their prediction accuracy.

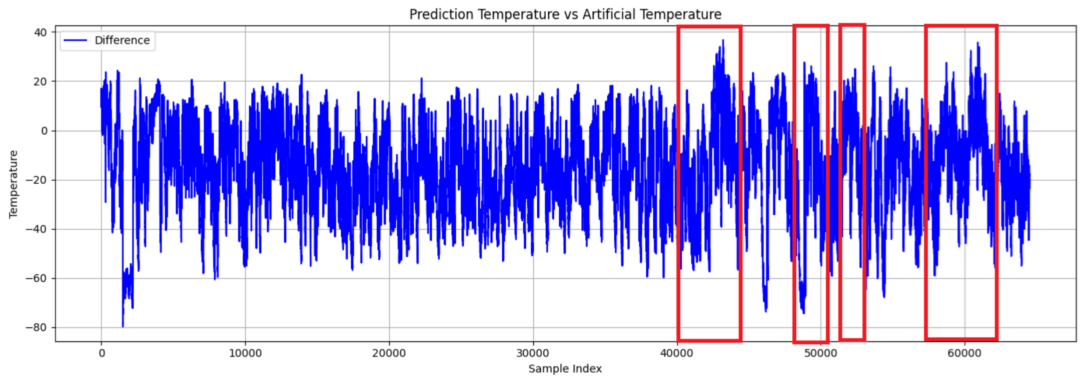

Figure 9: DSCNN_BILSTM residual graph between prediction temperature and artificial temperature

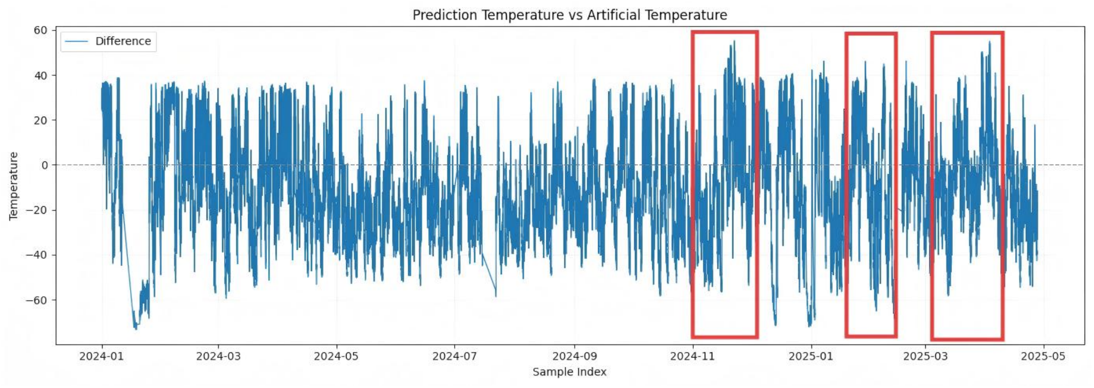

Figure 10: XGBoost residual graph between prediction temperature and artificial temperature

Figure 9 and 10 respectively display the prediction residuals for the DSCNN_BiLSTM and XGBoost models (in their optimal configurations) after injecting artificial temperature trends. The residuals represent the differences between the synthetic (manually altered) temperature and the model's predicted temperature. Both residual plots clearly highlight two distinct upward trend periods and two high-temperature peak regions, marked in red boxes. These patterns indicate that, despite the presence of anomalous trends, both models maintain a degree of predictive stability and robustness when confronted with such abnormal patterns.

5 Conclusions

In this project, we benchmarked a variety of modeling approaches—including statistical methods, gradient boosting, and deep learning architectures—for the task of temperature prediction in wind turbines based on SCADA data.
Among all models, the DSCNN-BiLSTM with attention achieved the best performance, reaching an R^2 score of approximately 0.940 across all test scenarios. This demonstrates the strong ability of hybrid deep learning models to capture both spatial and temporal

dependencies in complex operational data. In contrast, traditional models such as linear regression and XGBoost showed lower accuracy, particularly under dynamic or nonlinear conditions. A simple physics-informed model was also included as a baseline, offering interpretability and stability under healthy conditions.
Overall, our results highlight the effectiveness of deep learning—especially convolutional-recurrent architectures—in predictive maintenance tasks for wind turbines, offering high accuracy and strong potential for practical deployment.

5.1 Future Work

5.1.1 Multi-Turbine Data Preparation

The existing data preparation pipeline can be extended to extract operational telemetry from multiple turbines. To retain turbine-specific context, one-hot encoding of device IDs can be incorporated as an optional feature.
Cleaned datasets across turbines can be combined to create a unified multi-turbine dataset for testing and training. This would allow previously trained models to be evaluated on unseen turbines enabling the study of spatial generalization and model robustness.

5.1.2 Model Deployment and Monitoring

Future work should explore the deployment of trained models into real-time monitoring pipelines. This includes integrating models with SCADA systems for live prediction, adding alert thresholds for anomaly detection, and designing dashboards for condition monitoring. In addition, drift detection and retraining strategies should be investigated to ensure long-term model reliability in production environments.

5.1.3 Integration of Domain Knowledge

Further research could also explore improved integration of physical constraints and domain knowledge into deep learning models. For example, hybrid physics-informed neural networks (PINNs), constraint-aware loss functions, or causal modeling approaches may enhance model interpretability and safety—especially in mission-critical applications.

References

[1] Min Zhang et al. "Temperature Prediction and Fault Warning of High-Speed Shaft of Wind Turbine Gearbox Based on Hybrid Deep Learning Model". In: *Journal of Marine Science and Engineering* 13.7 (2025). ISSN: 2077-1312. DOI: `10.3390/jmse13071337`. URL: `https://www.mdpi.com/2077-1312/13/7/1337`.

[2] Min Zhang et al. "Temperature Prediction and Fault Warning of High-Speed Shaft of Wind Turbine Gearbox Based on Hybrid Deep Learning Model". In: *Journal of Marine Science and Engineering* 13 (July 2025), p. 1337. DOI: `10.3390/jmse13071337`.

[3] Hlaing Min Soe and Arkar Htet. "A Comprehensive Review of SCADA-Based Wind Turbine Performance and Reliability Modeling with Machine Learning Approaches". In: *Journal of Technology Innovations and Energy* 3.3 (2024), pp. 68–92.

[4] Wisdom Udo and Yar Muhammad. “Data-driven predictive maintenance of wind turbine based on SCADA data”. In: *IEEE Access* 9 (2021), pp. 162370–162388.

[5] Tianqi Chen and Carlos Guestrin. “XGBoost: A scalable tree boosting system”. In: *arXiv* (2016). arXiv:1603.02754.

[6] Jakob Schwerter et al. “Which Imputation Fits Which Feature Selection Method? A Survey-Based Simulation Study”. In: *arXiv preprint arXiv:2412.13570* (2024).

[7] Xavier Chesterman et al. “Overview of normal behavior modeling approaches for SCADA-based wind turbine condition monitoring demonstrated on data from operational wind farms”. In: *Wind Energy Science* 8.6 (2023), pp. 893–924.

[8] Timo Lichtenstein, Victor von Maltzahn, and Karoline Pelka. “Empirical Physics-Based Normal Behavior Modeling of Drive Train Temperatures With High-Resolution Wind Turbine Operating Data”. In: *Wind Energy* (2024), e2961.

[9] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[10] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. “The Performance of LSTM and BiLSTM in Forecasting Time Series”. In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 3285–3292. DOI: 10.1109/BigData47090.2019.9005997.

[11] Xinping Li et al. “A Hybrid DSCNN-BiLSTM Model for Accurate Wind Turbine Temperature Prediction”. In: *Processes* 13.4 (2025). ISSN: 2227-9717. DOI: 10.3390/pr13041143. URL: https://www.mdpi.com/2227-9717/13/4/1143.

[12] Liulin Yang et al. “Enhanced GAIN-Based Missing Data Imputation for a Wind Energy Farm SCADA System”. In: *Electronics* 14.8 (2025). ISSN: 2079-9292. DOI: 10.3390/electronics14081590.

[13] Rosemary Tawn, Jethro Browell, and Iain Dinwoodie. “Missing data in wind farm time series: Properties and effect on forecasts”. In: *Electric Power Systems Research* 189 (2020), p. 106640. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2020.106640.

[14] Mindaugas Jankauskas et al. “Exploring the Limits of Early Predictive Maintenance in Wind Turbines Applying an Anomaly Detection Technique”. In: *Sensors* 23.12 (2023). ISSN: 1424-8220. DOI: 10.3390/s23125695. URL: https://www.mdpi.com/1424-8220/23/12/5695.

[15] Dede Tarwidi et al. “An optimized XGBoost-based machine learning method for predicting wave run-up on a sloping beach”. In: *MethodsX* 10 (2023), p. 102119. DOI: 10.1016/j.mex.2023.102119. URL: https://www.sciencedirect.com/science/article/pii/S2215016123001206.

Appendix

Table 11: Performance of XGBoost (autoencoder features) on original and artificial test sets.

Model	**Feature**	**Test Set**	**RMSE**	$\mathbf{R^2}$	**MAE**	**MedAE**
XGboost	autoencoder	original	8.0469	0.7848	6.0008	4.4928
XGboost	autoencoder	artificial	26.570	-1.280	21.739	19.707

Table 12: Definition and Parameters of Network Layers for DSCNN-BiLSTM model

Definition of Layers	**Parameters of Layers**
Input layer	Size: $(1, n)$
DSCNN D1	Channel number: 32, kernel size: 3, padding: 1
Batch Normalization layer	-
Pooling layer	Kernel size: 2, stride: 2
DSCNN D2	Channel number: 64, kernel size: 3, padding: 1
Pooling layer	Kernel size: 2, stride: 2
Batch Normalization layer	-
BiLSTM	Hidden units: 64
Activation layer	Activation function: ReLU
Dropout layer	Dropout number: 0.3
Output layer	-