EUROPEAN CONSORTIUM FOR MATHEMATICS IN INDUSTRY

ECMI Modeling Week 2025
Kaunas University of Technology, Lithuania

Climate in the Light of Mathematical Equations

Dr. Davor Kumozec (University of Novi Sad)

Valeriia Baranivska (Igor Sikorsky Kyiv Polytechnic Institute)
Ilaria Astrid Bartsch (Università degli Studi di Milano)
Janne Finn Heibel (University of Koblenz)
Patrícia Marques (Instituto Superior Técnico)

05/07/2025

Contents

1 Introduction

The project focuses on climate-related fluid behavior and its variations over time. We aim to study atmospheric dynamics using two approaches: (1) a system of partial differential equations (PDEs) based on the Euler equations, and (2) a statistical learning method, specifically, a hybrid SARIMAX-LSTM model.

The simulation is based on the assumption of the evolution of a one-dimensional fluid with solar heating effects that mimic day-night temperature cycles. The governing system includes mass, momentum, and energy conservation laws, closed with an ideal gas equation of state. The simulation captures physical behaviors such as pressure waves, thermal variations, and momentum transport due to wind.

This document is structured as follows: Section 2 describes the PDEs method, in particular, outlines the theoretical foundation and equations used for the PDEs method and the numerical techniques applied to solve the equations, particularly the Lax–Friedrichs scheme. Section 3 presents the machine learning method, i.e., the hybrid SARIMAX-LSTM model. The discussion and conclusions could be found in Section 4, including comparison between methods and real data, as well as concluding remarks and a brief discussion of the approach's limitations and possible extensions.

2 PDEs Method

2.1 System of Equations

The fluid behavior is modeled using the Euler equations in one spatial dimension. These equations express the conservation of mass, momentum, and energy as

$$\begin{aligned} \rho_t + (\rho v)_x &= 0, && \text{(Mass conservation)} \\ (\rho v)_t + (\rho v^2 + p)_x &= 0, && \text{(Momentum conservation)} \\ E_t + ((E+p)v)_x &= 0. && \text{(Energy conservation)} \end{aligned}$$

The energy density E is composed of internal and kinetic energy

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2,$$

with ρ the air density, v the velocity, p the pressure, and γ the heat capacity ratio (typically $\gamma \approx 1.4$ for air).

In terms of momentum $m = \rho v$, the system becomes

$$\begin{aligned} \rho_t + m_x &= 0, \\ m_t + \left(\frac{m^2}{\rho} + p\right)_x &= 0, \\ E_t + \left((E+p)\frac{m}{\rho}\right)_x &= 0. \end{aligned} \tag{1}$$

A time-dependent source term $Q(t)$ is added to the energy equation to simulate daylight solar heating, defined as

$$Q(t) = Q_{\max} \cdot \sin(\omega t')\, \chi_{\text{day}}, \quad \chi_{\text{day}} = \begin{cases} 1, & t_{\text{start}} \le t \bmod T \le t_{\text{end}} \\ 0, & \text{otherwise} \end{cases}$$

where

$$t' = t \bmod T - t_{\text{start}}, \quad \omega = \frac{\pi}{t_{\text{end}} - t_{\text{start}}}$$

is the angular frequency ensuring a half-sine wave over the daylight interval $[t_{\text{start}}, t_{\text{end}}]$, and $T = 86400$ is the number of seconds in a day. The term $Q(t)$ models solar input with a smooth rise and fall during the day and zero heating at night.

The simulation outputs include the conserved quantities (ρ, m, E) and derived quantities pressure

$$p = (\gamma - 1)\left(E - \frac{m^2}{2\rho}\right)$$

and temperature

$$T = \frac{p}{\rho R}, \quad R = 287,$$

in Kelvin, from the ideal gas law. To convert to Celsius, substract 273.15.

2.2 Discretization

A nonlinear hyperbolic conservation law is defined through a flux function $f(u)$ as

$$u_t + f(u)_x = 0,$$

posed on the domain $b \leq x \leq c,\ 0 \leq t \leq d$, with initial and boundary conditions

$$u(x,0) = u_0(x), \quad \begin{matrix} u(b,t) = u_b(t), \\ u(c,t) = u_c(t). \end{matrix}$$

To solve this equation numerically, the domain $(b,c) \times (0,d)$ is discretized on a uniform grid with spacing Δx in space and Δt in time. The numerical solution $u_i^n \approx u(x_i, t^n)$ is defined at the grid points

$$x_i = b + i\Delta x, \quad t^n = n\Delta t, \quad \text{for } i = 0, \ldots, N, \quad n = 0, \ldots, M,$$

where

$$N = \frac{c-b}{\Delta x}, \quad M = \frac{d}{\Delta t}$$

represent the number of grid intervals. Boundary conditions $u_0^n = u_b(t^n)$ and $u_N^n = u_c(t^n)$ are enforced at each time step.

2.3 Numerical Scheme

To numerically solve the Euler system, we applied the Lax–Friedrichs finite difference method, which is explicit and suited for hyperbolic PDEs. This method was chosen because it is not computationally expensive and with first order of accuracy. For a conservative equation of the form

$$g_t + f(g)_x = 0,$$

the Lax–Friedrichs update formula is:

$$g_i^{n+1} = \frac{1}{2}(g_{i+1}^n + g_{i-1}^n) - \frac{\Delta t}{2\Delta x}(f(g_{i+1}^n) - f(g_{i-1}^n)).$$

This method is applied component-wise to the vector of conserved variables

$$U = \begin{bmatrix} \rho \\ m \\ E \end{bmatrix}_t, \quad f(U) = \begin{bmatrix} m \\ \left(\frac{3-\gamma}{2}\right)\frac{m^2}{\rho} + (\gamma-1)E \\ \left(\gamma E - (\gamma-1)\frac{m^2}{2\rho}\right)\frac{m}{\rho} \end{bmatrix}_x,$$

based on equation 2.1, and substituting p as

$$p = \left(E - \frac{m^2}{2\rho}\right)(\gamma - 1). \tag{2}$$

The time-step condition enforced to ensure numerical stability is

$$\Delta t \leq \frac{\Delta x}{s}, \quad s = \frac{\rho}{m} + \sqrt{\frac{\gamma p}{\rho}},$$

where $\sqrt{\frac{\gamma p}{\rho}}$ is the local speed of sound (7).

The treatment of the non-homogeneous part in the energy equation is done by using the fractional step method, where we first solve homogeneous part and after that $E_t = Q$ was solved using the forward Euler method.

The heating source term $Q(t)$ is explicitly added after the homogeneous update of the energy equation.

$$E_i^{n+1} = E_i^* + \Delta t \cdot Q(t)$$

where E_i^* is the result of the Lax-Friedrichs update without the source, and $\Delta t \cdot Q(t)$ is the contribution of external heating.

2.4 Data and Implementation

To compare the simulation used in the model with the actual data on the specified day, real data is used to initialize the variables ρ, m and E. For this, 23 weather stations across Lithuania are given and taken into account. These stations are approximately on a straight line connecting Klaipeda in the west with Vilnius in the southeast of Lithuania. All stations are roughly separated the same distance so an equidistant grid can be assumed. At these stations temperature T, air pressure p and wind speed v at midnight on 31 December are given.

To have realistic and meaningful boundary conditions, the mentioned data is also given in Klaipeda and Vilnius every hour on the specified day. In order to initialize the variables used in the code, the obtained data need to be transformed via certain formulas. To receive the density of the air the formula

$$\rho = \frac{p}{R \cdot T}$$

is used, where $R = 289.32 \, \frac{J}{mol \cdot K}$ denotes the ideal gas constant.
To obtain momentum m, the product of density ρ and wind speed u

$$m = \rho \cdot v$$

is used as seen above in this report.
The energy E is also obtained from data due to the equation

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2$$

which is mentioned earlier in this paper.

A transformation is applied to the wind speed v. Since the given values from the weather stations are absolute values of the speed, the direction of the wind also plays a role because only the part of the wind parallel to the line between Klaipeda and Vilnius can be taken into account in the model. For that, the angle of the connecting line and the wind is the deciding part. If the equator is considered to be a horizontal line with angle 0, then the connecting line between Klaipeda and Vilnius has an angle of $\theta = 22.5$. From the weather data it is known that on the specified day there was wind from the southwest, which means an angle of $\alpha = 45$. So for the overall angle between the wind and the connecting line an angle of $\beta = \alpha + \theta = 67.5$ is considered. The part of v that remains after projecting on the connecting line is obtained by the cosine of β. To use this in the model, the values of v are multiplied by the factor $\cos\beta \approx 0.3827$, which is assumed to be constant over time.

The simulation of the solar source term $Q(t)$ (solar power absorbed per unit volume) is adapted to the real times of sunrise and sunset of Kaunas. The data show that sunrise is at 8:49 am and sunset is at 16:07 pm. In addition, the maximum solar power absorbed per unit volume Q_{max} is chosen based on a real value of energy source power density ($50\ W/m^2$), and considering a gas weight of $111m$, leading to $Q_{max} = 0.45W/m^3$.

All data, transformed and directly obtained by the station, are stored in an Excel file. The data needed in the code are then loaded into Python to do further work on it.

3 Hybrid SARIMAX-LSTM Model

Most machine learning (ML) algorithms perform well on specific tasks or datasets but struggle to generalize. Hybrid Machine Learning (HML) combines multiple models to overcome this limitation. In this project, a SARIMAX-LSTM hybrid was used to improve time series forecasting. SARIMAX captures linear trends and seasonality, while LSTM models the non-linear residuals.

3.1 Autoregressive processes

Autoregression (AR) is a method used to predict time series data by using its past values. It assumes that the current value depends on previous ones in a linear way. It's simple, effective, and forms the base for more advanced models like SARIMAX.

An autoregressive model of order p, denoted as AR(p), is defined as

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t,$$

where

X_t : Value of the time series at time t.

c : Constant term (intercept).

$\phi_1, \phi_2, \ldots, \phi_p$: Model parameters (coefficients).

p : Order of the model (number of lagged terms).

ε_t : White noise (random error, normally distributed with mean 0 and constant variance).

The simplest autoregressive model is AR(1), where $p = 1$,

$$X_t = c + \phi_1 X_{t-1} + \varepsilon_t.$$

For an AR(1) model to be stationary (i.e., its statistical properties like mean and variance do not change over time), then,

$$|\phi_1| < 1 \tag{3}$$

More generally, for an AR(p) model to be stationary, the roots of the characteristic equation must lay outside the unit circle in the complex plane.

3.2 SARIMAX Model

The Seasonal Autoregressive Integrated Moving Average with Exogenous Regressors (SARIMAX) extends ARIMA framework - which combines autoregressive (AR), differencing (I), and moving average (MA) components — by modeling seasonality and incorporating external variables. This allows SARIMAX to handle time series data that exhibit recurring seasonal patterns and are influenced by exogenous factors.

The model is denoted as SARIMAX$(p, d, q)(P, D, Q)_m$, where

- (p, d, q): Non-seasonal AR order, differencing, and MA order.
- (P, D, Q): Seasonal AR order, seasonal differencing, and seasonal MA order.
- m: Length of the seasonal period (e.g., 12 for monthly data).
- X: Exogenous variables.

Using the backshift operator B ($By_t = y_{t-1}$), SARIMAX can be written as

$$\Phi_P(B^m)\phi_p(B)(1 - B^m)^D(1 - B)^d y_t = \sum_{i=1}^{k} \beta_i x_{i,t} + \Theta_Q(B^m)\theta_q(B)\varepsilon_t,$$

where

y_t**:** Time series value at time t.

$\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$**:** Non-seasonal AR operator.

$\theta_q(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$**:** Non-seasonal MA operator.

$(1 - B)^d$**:** Non-seasonal differencing.

$\Phi_P(B^m) = 1 - \Phi_1 B^m - \cdots - \Phi_P B^{Pm}$**:** Seasonal AR operator.

$\Theta_Q(B^m) = 1 + \Theta_1 B^m - \cdots + \Theta_Q B^{Qm}$**:** Seasonal MA operator.

$(1 - B^m)^D$**:** Seasonal differencing.

$x_{i,t}$**:** i-th exogenous input; β_i its coefficient.

ε_t**:** White noise error term.

Key features include differencing orders (d, D) to stabilize trends and seasonality, seasonal AR/MA terms (P, Q) to model recurring patterns, and exogenous regressors X to account for external influences, improving forecast accuracy.

3.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of neural network designed for time series data. Unlike basic RNNs, it can remember important information from earlier in the sequence, improving predictions.

Each LSTM cell maintains a memory state that carries key information forward. Three gate - **input**, **forget**, and **output** - control what to keep, update, or discard, based on the previous hidden state h_{t-1} and the current input x_t.

- **Forget Gate (f_t):** This gate decides what information to discard from the previous cell state, C_{t-1}. It passes h_{t-1} and x_t through a sigmoid function, which outputs a number between 0 ("completely forget this") and 1 ("completely keep this").
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
- **Input Gate (i_t):** This gate determines what new information to store in the cell state. It consists of two parts: a sigmoid layer that decides which values to update (i_t), and a 'tanh' layer that creates a vector of new candidate values, $\tilde{C}_t$.
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
- **Cell State Update:** The old cell state C_{t-1} is updated to the new cell state C_t. This is done by multiplying the old state by the forget gate's output (f_t) and then adding the product of the input gate's output and the candidate values ($i_t \odot \tilde{C}_t$).
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
- **Output Gate (o_t):** This gate decides what information to output from the cell state. The output will be a filtered version of the cell state. First, a sigmoid layer decides which parts of the cell state to output. Then, the cell state is passed through a 'tanh' function and multiplied by the output of the sigmoid gate.[12, 16]
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

In these equations, W and b represent the weight matrices and bias vectors for each respective gate, and $\odot$ denotes element-wise multiplication. This gated mechanism allows LSTMs to effectively learn and remember patterns over long sequences, making them a robust tool for complex time series analysis.

3.4 Hybrid Model

Generating future forecasts with the hybrid model is an iterative, multi-step process that leverages both the SARIMAX and LSTM components to predict the next 24 hourly temperature values.

The first step is **1. forecasting the linear Component of the time series with SARIMAX**. The trained SARIMAX model is used to predict the next 24 hours. Since the model relies on external variables (such as humidity and pressure), these are assumed to remain constant at their last observed values during the forecast period. The result is a series of 24 predicted values capturing the baseline linear trend and seasonal patterns of the temperature.

The second step is **2. iterative forecasting of the non-Linear component with LSTM**, i.e., predict the non-linear residuals that the SARIMAX model cannot capture. This is performed in a **step-by-step loop for each of the 24 future hours**:

1. **Initialisation:** Start with the most recent input data from the historical records.
2. **Iterative Prediction Loop (for each of the 24 hours):**
   (a) Predict **residual error** for the next hour using the current input sequence.
   (b) Convert the predicted residual back to the original **temperature scale**.
   (c) **Combine Forecasts:** Add the predicted residual to the SARIMAX forecast for that hour to get the final temperature prediction.
   (d) **Update Input Sequence:** Remove the oldest data point and add the new predicted value to the input sequence. This updated sequence is used for the next hour's prediction.

This iterative combination ensures that each hourly forecast benefits from both the stable, long-term pattern recognition of the SARIMAX model and the nuanced, non-linear adjustments provided by the LSTM network.

3.5 Data and Implementation

To ensure the model is based on accurate and relevant real-world conditions, a systematic data acquisition process was implemented. The primary source for all meteorological data was the official Meteo.lt Application Programming Interface (API), provided by the Lithuanian Hydrometeorological Service (LHMT). This API offers direct and reliable access to historical observation data from its network of stations across Lithuania.

The primary goal is to develop an accurate model for forecasting the **target variable** hourly `airTemperature` at the `kauno-ams` meteorological station.

The dataset consists of hourly meteorological readings from three automated stations: Kaunas, Vilnius, and Klaipėda. It covers the period from December 1 to December 31 for each year between 2015 and 2024. Data was retrieved from the official `Meteo.lt` source using its Python API. [1]

The model uses several exogenous input variables, which are continuous meteorological features collected from all three stations:

- `airTemperature` $[C]$ - Air temperature
- `windSpeed` $[m/s]$ — Average wind speed
- `windGust` $[m/s]$ — Maximum wind gust
- `seaLevelPressure` $[Pa]$ — Sea-level atmospheric pressure
- `relativeHumidity` [%] — Relative humidity
- `cloudCover` [%] — Cloud cover
- `conditionCode` — Categorical variable for weather conditions (clear, rain, snow)

A thorough preprocessing and feature engineering pipeline was applied to prepare the raw meteorological data for modeling, ensuring quality and suitability for both SARIMAX and LSTM models.

The dataset was **loaded** from CSV, with the $observationTimeUtc$ column converted to datetime and set as the index. Data was filtered to include only December records from each year and limited to the target and input stations to focus on winter patterns. **Missing values** were filled using a two-step approach: forward-fill followed by backward-fill, ensuring all gaps were addressed.

To enhance model input, cyclical temporal features were created by encoding the hour of the day into sine and cosine components ($hour\_sin$, $hour\_cos$), capturing the 24-hour cycle. Similarly, wind direction was transformed using sine and cosine to preserve its circular nature:

$$wind_{sin} = \sin\left(windDirection \times \frac{\pi}{180}\right) \tag{4}$$

$$wind_{cos} = \cos\left(windDirection \times \frac{\pi}{180}\right) \tag{5}$$

[1] Meteo.lt API (Application programming interface) enables you to receive and use publicly available data measured by hydrological and meteorological stations of the Lithuanian Hydrometeorological Service under the Ministry of the Environment (hereinafter referred to as LHMT) and weather forecasts. Using Meteo.lt API , you can integrate hydrometeorological data provided by LHMT into your own applications.Link for API

To provide the model with explicit **spatial context**, the geographical distance between the target station (‘$kauno - ams$‘) and each of the input stations (‘$vilniaus - ams$‘, ‘$klaipedos - ams$‘) was calculated. Using the Haversine formula:

$$\text{hav}(\theta) = \text{hav}(\Delta\phi) + \cos(\phi_1)\cos(\phi_2)\text{hav}(\Delta\lambda) \tag{6}$$

where

$[\phi_1, \phi_2]$ are the latitudes of point 1 and point 2.

$[\lambda_1, \lambda_2$ are the longitudes of point 1 and point 2.

$[\Delta\phi = \phi_2 - \phi_1, \quad \Delta\lambda = \lambda_2 - \lambda_1]$.

which computes the great-circle distance between two points on a sphere, a static distance feature was created for each input station (e.g., ‘$dist\_kauno{-}ams\_to\_vilniaus{-}ams$‘). This feature gives the model a constant, quantitative measure of the spatial separation between the data sources.

The ‘*conditionCode*‘ feature is categorical, representing distinct weather conditions (e.g., ’*clear*’, ’*rain*’, ’*snow*’). To use this nominal data in the models, **one-hot encoding** was applied. This process creates new binary (0 or 1) columns for each unique category within the ‘*conditionCode*‘ feature. This prevents the model from incorrectly assuming an ordinal relationship between the different weather conditions and allows it to treat each condition as a distinct input.

After these steps, the final feature set for modeling was comprehensive, including the original numerical features, the engineered temporal and directional cyclical features, the static spatial distance features, and the binary columns from the one-hot encoded weather conditions.

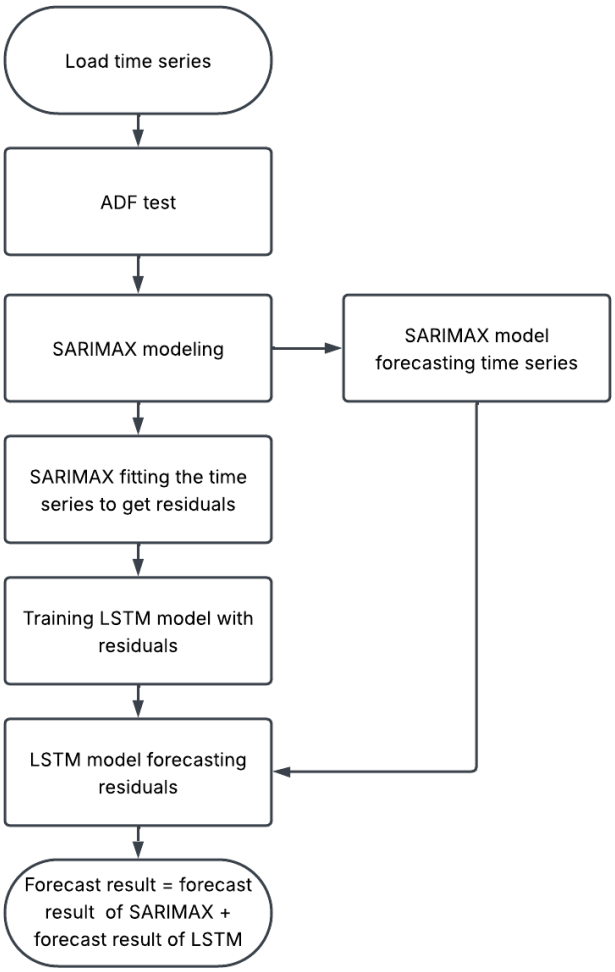

Figure 1: Hybrid model approach flowchart.

4 Results and Conclusions

4.1 Model Comparison

The **PDE-based model** produces physically plausible predictions for temperature, momentum density, energy density, and pressure. Temporal patterns follow expected meteorological behavior. For instance, the solar radiation peaks around 12 PM, resulting in a daytime increase in temperature and pressure, followed by a nighttime decline.

According to the ideal gas law, density ρ is inversely related to temperature T under constant pressure. Thus, as **temperature increases**, **density decreases**, which is consistent with physical intuition. Energy density varies mainly with temperature, while momentum density responds to changes in wind velocity and direction.

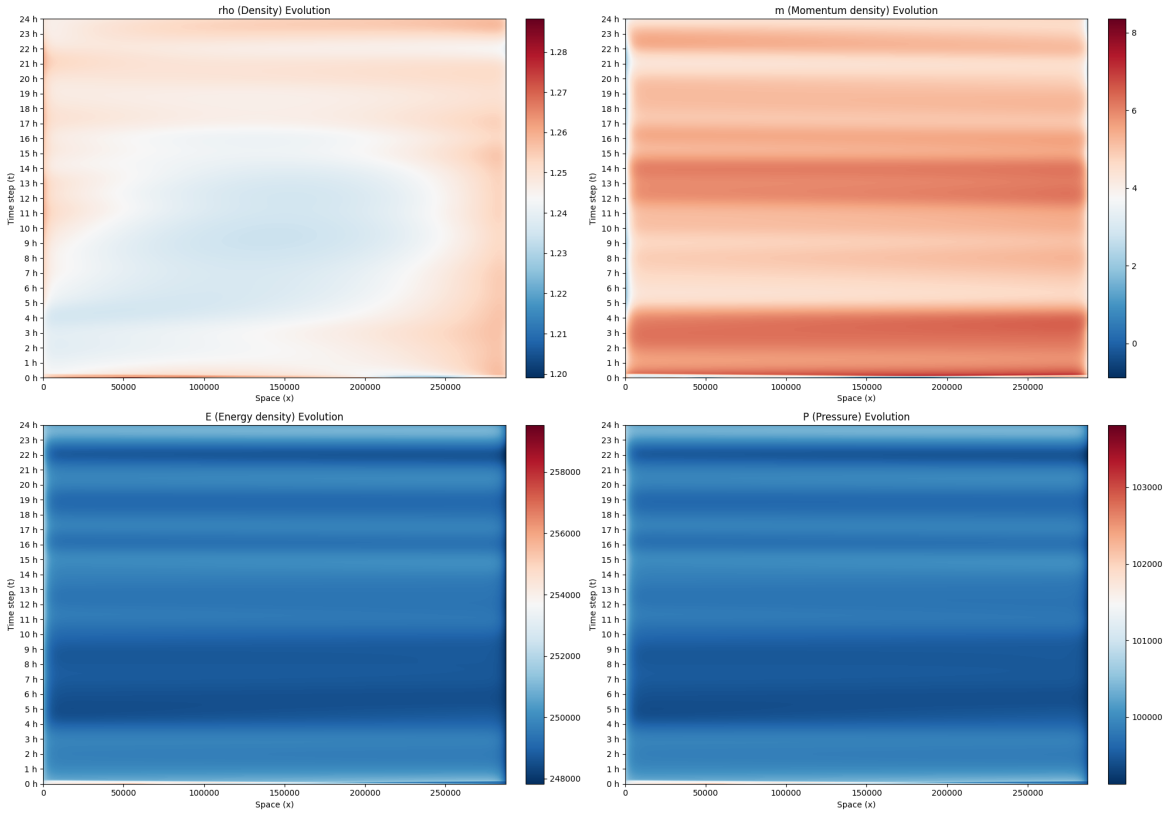

Figure 2: Simulation of the time evolution of physical variables across space on 31 December 2024: density (ρ) [kg/m^3], momentum density (m) [kg/(m^2·s)], energy density (E) [J/m^3], and pressure (P) [Pa]. The spatial domain spans from Klaipėda (left boundary) to Vilnius (right boundary).

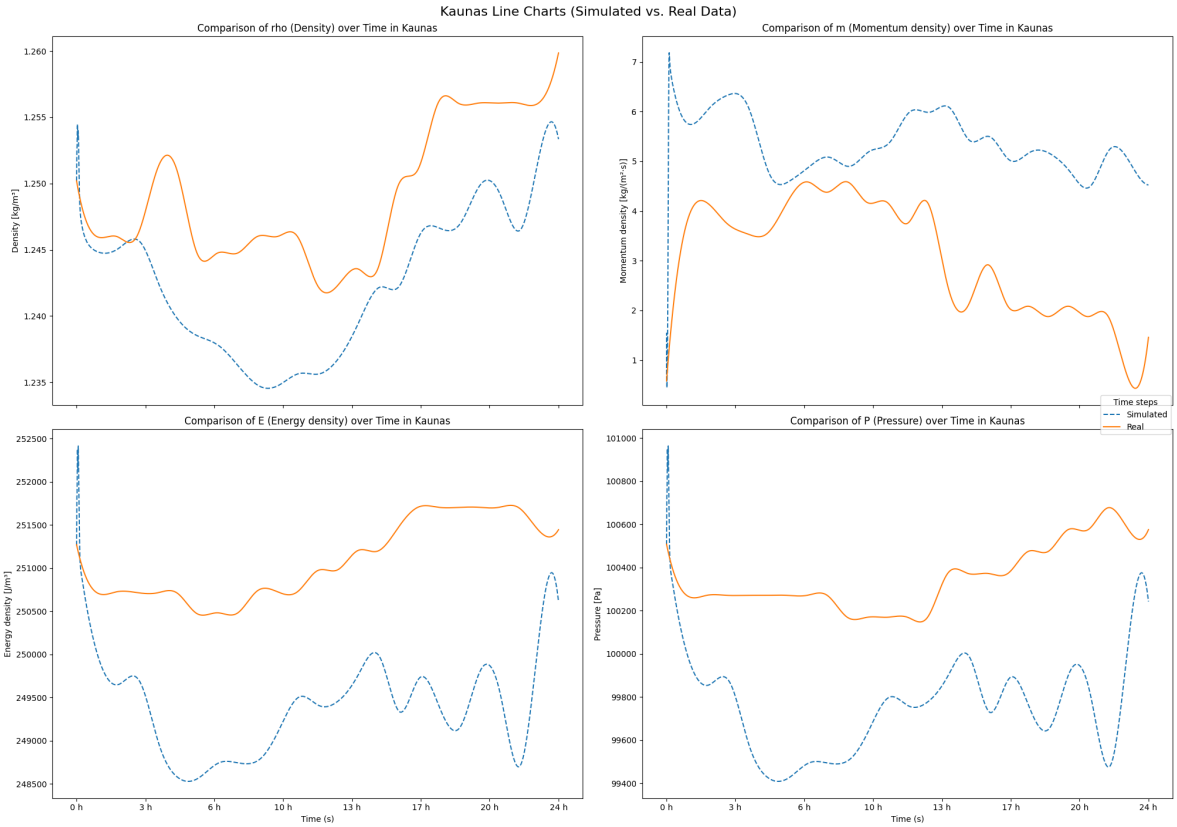

Figure 3: Comparison of model testing results with real data and predicted data. Date that exists in the real data records (31 December 2024)

The model predicts a sharp temperature peak around 12 PM, reaching around 6°C. The actual recorded data shows a more gradual increase, with a similar maximum of about 6°C. This indicates that the model estimates fairly the midday heating, likely due to a reasonable solar source forcing.

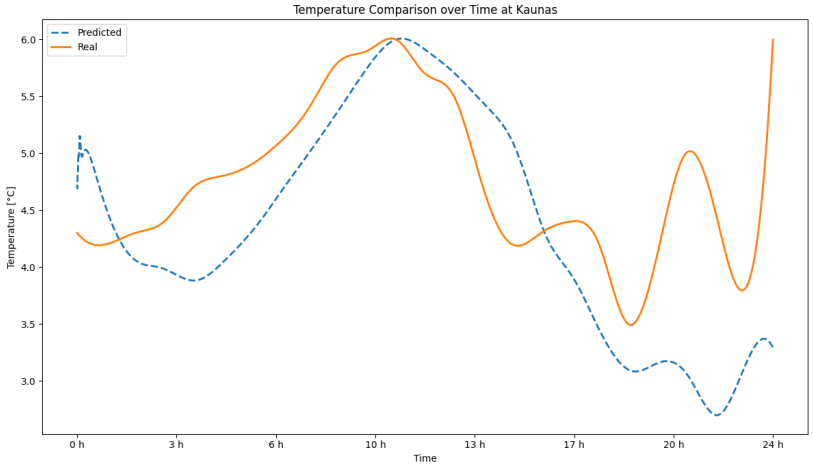

Figure 4: Comparison between simulated temperature in Kaunas from the PDE-based model (blue dashed line) and real recorded data (orange solid line), interpolated from hourly measurements to match simulation time steps (31 December 2024).

The tables related to the results from **Hybrid SARIMAX-LSTM Method** the with all the results are present in the Appendix.

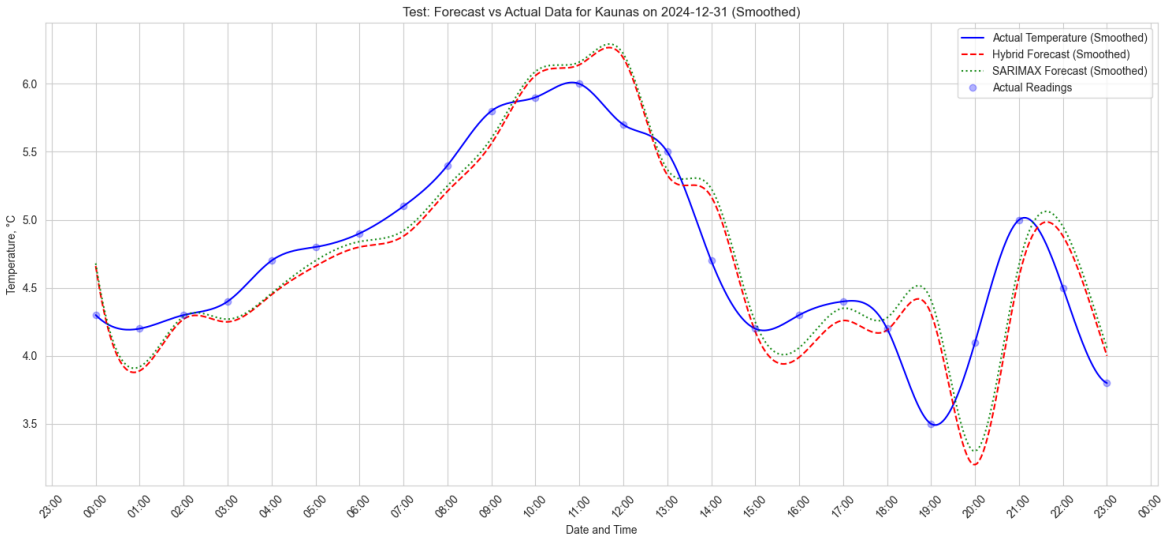

Figure 5: Comparison of model testing results with real data and predicted data. Date that exists in the real data records (31 December 2024)

Figure 5 presents a visual comparison of the model's predictions against the actual temperature data for December 31, 2024, a day within the test set. The blue line represents the ground truth temperature. The green dotted line shows the forecast from the SARIMAX model alone, while the red dotted line shows the final forecast from the complete hybrid model. It is visually evident that the hybrid model's predictions (red line) track the actual data (blue line) more closely than the SARIMAX-only predictions (green line), demonstrating the value added by the LSTM component in correcting for non-linear errors. The overall prediction error appears minimal, visually estimated at 0.3-0.5 degrees in most places.

The SARIMAX component effectively captured the underlying trends and seasonalities in the data, while the LSTM network proved adept at modeling the complex, non-linear relationships present in the SARIMAX residuals. The combined model demonstrated a high degree of accuracy on the unseen test data, as evidenced by the strong performance metrics. An R-squared value of 0.7057 indicates that the model explains over 70.5% of the temperature variance, and low RMSE (0.3638) and MAE (0.3063) values confirm that the predictions are, on average, very close to the actual values.

Visual diagnostics, including the forecast plots and heat map, further corroborated the model's effectiveness, showing a tight fit between predicted and actual values. The project illustrates the power of hybrid modeling in time series analysis. By decomposing the forecasting problem and applying the appropriate tool to each sub-problem, the SARIMAX-LSTM model overcomes the individual limitations of its components, resulting in a robust, accurate, and highly effective forecasting system.

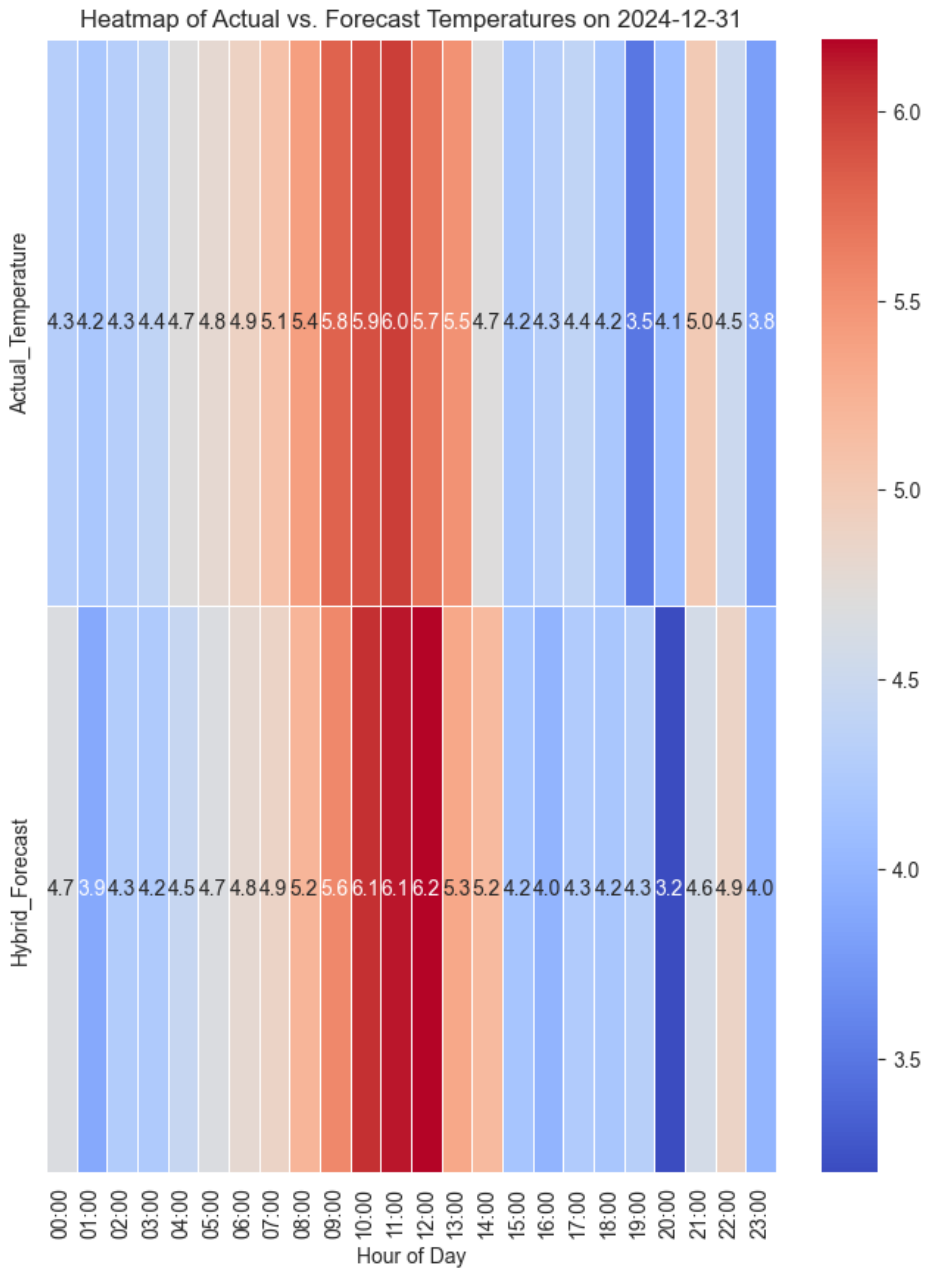

Figure 6: Heat map for comparison of model testing results with real data and predicted data (31 December 2024). The colour scale displays the temperature values from the smallest (light blue) to the largest (dark red). The close correspondence in colour between the "Actual" and "Predicted" rows for each hour provides a clear and intuitive visualisation of the model's high accuracy, as both rows display nearly identical colour patterns across the entire day.

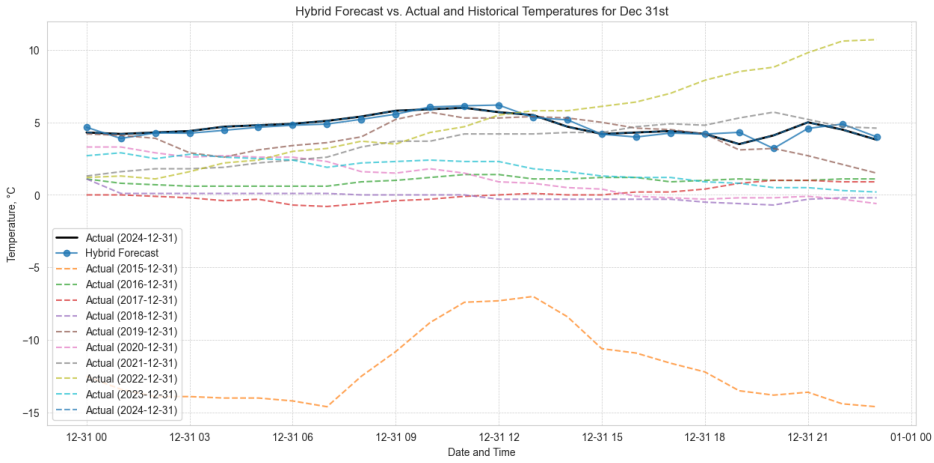

Figure 7: Comparison of the years (31 December 2024).

Figure 7 shows a comparison of the hybrid forecast with actual and historical temperatures as of December 31, 2024. Of particular interest are the anomalous years like 2015 and 2020.

2015 anomalies (orange line): shows a dramatic drop in temperature at the end of the period, which seems paradoxical since 2015 was the warmest year on record, beating the previous record of 2014 by 0.13°-0.18°C.

2020 anomalies (dark blue line): shows consistently high temperatures, which is supported by the scientific data: the global average surface temperature in 2020 tied 2016 as the warmest year on record.

This graph illustrates the complexity of climate prediction, where long-term warming trends are combined with short-term natural variations. Therefore, for future research, this may become one of the vectors of development – research and prediction of global warming (such "jumps").

It was successfully developed and evaluated a hybrid SARIMAX-LSTM model for time series forecasting of hourly temperature data. The underlying hypothesis was that a hybrid approach, leveraging a statistical model for linear patterns and a neural network for non-linear dynamics, would yield superior performance compared to either model in isolation.

4.2 Future Work

The **PDEs method** was expected to not fully capture the complexity of climate characteristics prediction. Part of it due to the simplicity of time and space updates, and part of it due to improvements to be done.

- **Averaging over multiple days:** Repeat the weather variable simulation for several days surrounding the target date, and collect the corresponding real data for those days. Then, compute the mean values for each location to enable a more robust and reliable comparison.
- **Incorporate source terms:** The current assumption of strict momentum and mass conservation may be overly simplistic, especially as it is an open/dynamic system. Consider introducing source terms to account for fluctuations in velocity and mass inflow/outflow.
- **Considering the three-dimensional nature of the problem:** In the present study, we have assumed one-dimensional variables. Nevertheless, given that we are modeling atmospheric phenomena, the full three-dimensional structure of the problem should be incorporated in future developments.
- **Enhancement of the numerical scheme:** For future developments, the Godunov method could be considered in order to improve the accuracy of the numerical approximation.

While the current **hybrid SARIMAX-LSTM model** performs well, several avenues for future improvement exist:

- **Expanded Exogenous Variables:** Incorporating additional meteorological data, such as cloud cover, solar radiation, or precipitation, could capture more of the variance and improve forecast accuracy.
- **Advanced Hyperparameter Tuning:** The current model's hyperparameters were selected based on standard heuristics. Employing more sophisticated optimisation techniques, such as Bayesian Optimisation, could lead to a more finely-tuned and potentially more accurate model.
- **Alternative Neural Network Architectures:** While LSTM is effective, other advanced recurrent architectures like the Gated Recurrent Unit (GRU) or models incorporating attention mechanisms could be explored to see if they offer improved performance or computational efficiency in modelling the residuals.
- **Rolling Forecast Evaluation:** A more rigorous evaluation could be performed using a rolling forecast origin. This would provide a better assessment of the model's stability and performance over different time periods, simulating a real-world deployment scenario more closely.

5 Group work dynamics

The group organized the work according to each member's strengths and expertise. One member concentrated on developing the machine learning component, focusing on model implementation. Two members took responsibility for the PDE part: one focused on deriving and formulating the mathematical equations, while the other worked primarily on coding and implementing these PDE models. Another member handled data collection and cleaning, ensuring the dataset was well-prepared for the modeling approach.

Throughout the project, although tasks were divided, all key steps and decisions were discussed collaboratively. This ensured that every part of the project was aligned with the group's objectives and integrated smoothly. The combination of specialized roles and regular communication helped maintain balance and progress in both the modeling and data preparation phases.

6 Instructor's assessment

The group was highly dedicated to solving the problem. When the issue was first presented, a model based on the Euler system was developed as a potential solution. After initial testing, it became clear that the model could be improved. A modification of the original problem was introduced to also account for the time of day—specifically, whether there was sunlight.

Given the variety of approaches explored, an additional model based on machine learning was also proposed, and the results were compared.

In conclusion, I am very pleased with my group's work, especially their dedication and teamwork.

Appendix

Metric	Value
RMSE	0.350500
MAE	0.274500
R-squared	0.726900
MAPE (%)	6.111800
Max Error	0.901300

Table 1: Metrics for assessing model quality

Description: This table summarises the performance of the hybrid SARIMAX-LSTM model on the test dataset. Each metric provides a different perspective on the model's accuracy:

- **RMSE** (Root Mean Square Error): Measures the square root of the average of squared differences between predicted and actual values. The value indicates that, on average, the model's predictions are about 0.35 degrees from the actual temperature, with larger errors being penalised more heavily.
- **MAE** (Mean Absolute Error): Represents the average absolute difference between the predicted and actual values. The MAE of 0.27 suggests the average prediction error is approximately 0.30 degrees.
- **R-squared**: The coefficient of determination indicates the proportion of the variance in the dependent variable (temperature) that is predictable from the independent variables. That means that over 72% of the variability in temperature is explained by the model, signifying an excellent fit.
- **MAPE** (Mean Absolute Percentage Error): Expresses the average error as a percentage of the actual values. The MAPE of 6.11% is relatively low, which is likely due to the stability of winter weather patterns and the model's effective use of multiple data sources.
- **Max Error**: This metric captures the single worst-case prediction error. That means the largest single deviation between a predicted and an actual temperature was approximately 0.9 degrees.

Description: This table provides a detailed, hour-by-hour breakdown of the hybrid model's performance on a specific day from the test set. It clearly illustrates the mechanism of the hybrid approach. The $SARIMAX\_Forecast$ column shows the prediction from the linear model, while the $LSTM\_Residual\_Forecast$ column shows the prediction of the non-linear error component. The final $Hybrid\_Forecast$ is the sum of these two columns, demonstrating how the LSTM corrects the initial SARIMAX prediction to produce a more accurate final result that is closer to the $Actual\_Temperature$.

Date_time	Actual_Temperature	SARIMAX_Forecast	LSTM_Residual_Forecast	Hybrid_Forecast
2024-12-31 00:00:00	4.300000	4.680000	-0.020000	4.660000
2024-12-31 01:00:00	4.200000	3.920000	-0.030000	3.890000
2024-12-31 02:00:00	4.300000	4.290000	-0.010000	4.270000
2024-12-31 03:00:00	4.400000	4.270000	-0.020000	4.250000
2024-12-31 04:00:00	4.700000	4.460000	-0.010000	4.450000
2024-12-31 05:00:00	4.800000	4.700000	-0.040000	4.660000
2024-12-31 06:00:00	4.900000	4.840000	-0.040000	4.800000
2024-12-31 07:00:00	5.100000	4.920000	-0.040000	4.880000
2024-12-31 08:00:00	5.400000	5.250000	-0.040000	5.210000
2024-12-31 09:00:00	5.800000	5.600000	-0.040000	5.560000
2024-12-31 10:00:00	5.900000	6.090000	-0.030000	6.060000
2024-12-31 11:00:00	6.000000	6.160000	-0.030000	6.140000
2024-12-31 12:00:00	5.700000	6.220000	-0.030000	6.190000
2024-12-31 13:00:00	5.500000	5.370000	-0.040000	5.330000
2024-12-31 14:00:00	4.700000	5.230000	-0.060000	5.170000
2024-12-31 15:00:00	4.200000	4.250000	-0.070000	4.180000
2024-12-31 16:00:00	4.300000	4.060000	-0.070000	3.990000
2024-12-31 17:00:00	4.400000	4.350000	-0.090000	4.260000
2024-12-31 18:00:00	4.200000	4.280000	-0.090000	4.190000
2024-12-31 19:00:00	3.500000	4.410000	-0.100000	4.310000
2024-12-31 20:00:00	4.100000	3.300000	-0.100000	3.200000
2024-12-31 21:00:00	5.000000	4.670000	-0.090000	4.580000
2024-12-31 22:00:00	4.500000	4.950000	-0.080000	4.880000
2024-12-31 23:00:00	3.800000	4.060000	-0.070000	4.000000

Table 2: Comparison of model testing results with real data and predicted data. Date that exists in the real data records (15 December 2023)

Table 3 demonstrates the primary output and practical application of the developed model. It presents a 24-hour, out-of-sample forecast for a future date, December 31, 2025. Each row provides the predicted temperature for a specific hour, showcasing the model's capability to generate actionable, long-range forecasts.

Date and time	Forecast Temperature for 2025	Forecast Temperature for 2024
2025-12-31 00:00:00	3.59	4.3
2025-12-31 01:00:00	3.57	4.2
2025-12-31 02:00:00	3.55	4.3
2025-12-31 03:00:00	3.49	4.4
2025-12-31 04:00:00	3.51	4.7
2025-12-31 05:00:00	3.50	4.8
2025-12-31 06:00:00	3.51	4.9
2025-12-31 07:00:00	3.52	5.1
2025-12-31 08:00:00	3.61	5.4
2025-12-31 09:00:00	3.70	5.8
2025-12-31 10:00:00	3.79	5.9
2025-12-31 11:00:00	3.88	6.0
2025-12-31 12:00:00	3.89	5.7
2025-12-31 13:00:00	3.87	5.5
2025-12-31 14:00:00	3.82	4.7
2025-12-31 15:00:00	3.75	4.2
2025-12-31 16:00:00	3.68	4.3
2025-12-31 17:00:00	3.71	4.4
2025-12-31 18:00:00	3.72	4.2
2025-12-31 19:00:00	3.75	3.5
2025-12-31 20:00:00	3.74	4.1
2025-12-31 21:00:00	3.74	5.0
2025-12-31 22:00:00	3.68	4.5
2025-12-31 23:00:00	3.67	3.8

Table 3: Hourly forecast for 31 December 2025

References

[1] Brownlee, J. (2021, May 27). *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. Machine Learning Mastery. Retrieved from [Link]

[2] Dinpashoh, Y., Miraki, A., Yasi, Y., & Farhadi, N. (2024). Predicting climate change impacts on water quality by developing a hybrid autoregressive-long short-term memory model. *Frontiers in Environmental Science, 12*. [Link]

[3] Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts. Retrieved from [Link]

[4] Zhang, Y., Liu, L., Wang, Y., Li, J., & Zhao, Y. (2020). A Novel Hybrid Model for Stock Price Forecasting based on ARIMA and LSTM. In *2020 International Conference on Computer Information and Big Data Applications (CIBDA)* (pp. 163-167). IEEE. [Link]

[5] Hansen, B.E. (2022) *Econometrics*. Princeton: Princeton University Press. [1, 2]

[6] Stock, J.H. and Watson, M.W. (2015) *Introduction to Econometrics*. 3rd edn., Updated. Boston: Pearson. [3]

[7] Randall J. LeVeque *Numerical Methods for Conservation Laws*. Lectures in Mathematics, ETH Zurich, Department of Mathematics, Research Institute of Mathematics. Second Edition