Forecasting Ambient Temperature with RNNs: A Comparison of Linear Models and Sequence Networks on the Jena Climate Dataset

Mintesnot Kassa
Department of Computer and Information Sciences
Fordham University
mak29@fordham.edu

***Abstract*—Accurate short horizon ambient temperature forecasts are important for residential comfort, energy management, and operational decision making. Weather time series exhibit complex structure: strong diurnal and seasonal patterns, autocorrelation, and nonlinear interactions between variables such as humidity, pressure, and wind. Traditional linear models can be competitive when equipped with carefully engineered lag features, yet they struggle to capture long-range temporal dependencies. In this project, we use the Jena Climate time series dataset (10-minute observations collected in Jena, Germany) to compare strong classical baselines persistence, Linear Regression, Ridge Regression against recurrent neural networks (LSTM and GRU) trained directly on historical multivariate sequences. We develop an end-to-end pipeline including cleaning, normalization, exploratory data analysis (EDA), supervised window generation, and leakage safe time aware evaluation. We consider multiple forecast horizons (1, 2, and 6 hours ahead) and report root mean squared error (RMSE) and coefficient of determination (R^2). The results illustrate when sequence models provide meaningful improvements over linear models, and when simple baselines are already strong. In addition, the project serves as a compact case study of how to design, train, and evaluate deep sequence models on a real-world time series in a way that avoids common pitfalls such as temporal leakage.**

***Index Terms*—Time series forecasting, Jena Climate, ambient temperature, LSTM, GRU, Ridge Regression, RNN.**

I. Introduction

Short horizon temperature forecasting underpins many operational tasks, from heating, ventilation, and air conditioning (HVAC) control to renewable energy integration and outdoor activity planning. For many of these applications, a forecast horizon of a few hours is sufficient for decision making, but high accuracy and robust uncertainty behavior are essential.

Ambient temperature behaves as a typical meteorological time series. It exhibits strong diurnal seasonality, slower seasonal trends, and autocorrelation at multiple lags. Other variables such as air pressure, humidity, water vapor content, and wind speed are correlated with temperature and can serve as useful covariates. Modeling such data raises the question of how much benefit is obtained from modern sequence models (e.g., recurrent neural networks) compared to classical regression methods.

Beyond pure forecast accuracy, there is also a systems perspective: a forecast model needs to be sufficiently simple to deploy, robust to missing data, and fast enough to run in near real time. Linear models score well on simplicity and interpretability, whereas RNNs are more flexible but are often treated as black-box models and may require more careful hyperparameter tuning and monitoring.

The goal of this project is to quantify the value of recurrent neural networks (RNNs) for short horizon temperature forecasting on the Jena Climate dataset. Concretely, we:

- Predict near term ambient temperature (1, 2, and 6 hours ahead) using recent multivariate weather history sampled every 10 minutes.
- Compare RNNs (LSTM and GRU) against strong linear baselines (Persistence, Linear Regression, Ridge Regression on lagged features).
- Use leakage-safe time aware evaluation: blocked or rolling time-series splits rather than random cross-validation.
- Provide interpretable EDA and diagnostics to illustrate seasonality, correlations, and error patterns.
- Relate our empirical findings to recent research on deep learning for time series and weather forecasting.

The experiments and implementation are inspired by the public Keras and Kaggle examples for the Jena Climate dataset, but implemented as an end-to-end pipeline tailored to this course project, with additional baselines and evaluation details that reflect best practices in modern time-series forecasting.

II. Dataset

A. Jena Climate Time-Series

The Jena Climate dataset consists of multivariate meteorological measurements recorded at a weather station near the Max Planck Institute for Biogeochemistry in Jena, Germany. The publicly available version on Kaggle comprises measurements from 2009 to 2016 with a sampling interval of 10 minutes. Each row corresponds to a timestamp and includes, among others:

- Ambient temperature T in degrees Celsius (`T (degC)`),
- Air pressure (`p (mbar)`),
- Dew point temperature (`Tdew (degC)`),
- Relative humidity (`rh (%)`),
- Water vapor pressure and concentration,

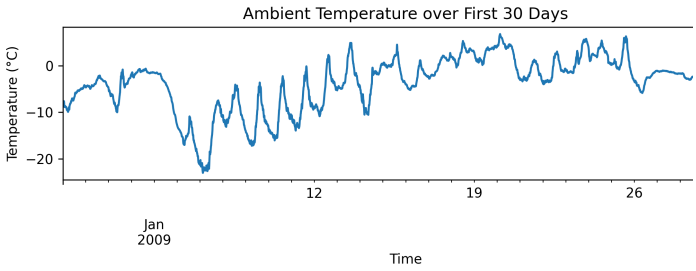

Fig. 1. Ambient temperature (degC) for the first month of 2009. Clear diurnal cycles and slower trends are visible.

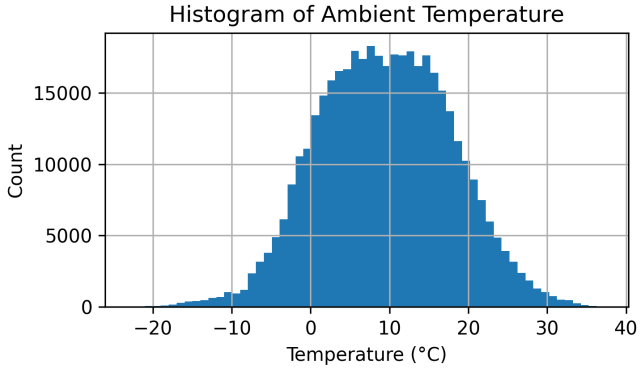

Fig. 2. Histogram of ambient temperature over the full dataset.

- Air density (`rho (g/m**3)`),
- Wind speed and maximum gust (`wv (m/s)`, `max. wv (m/s)`),
- Wind direction (`wd (deg)`).

This study focus on forecasting the ambient temperature series, using all available variables as covariates. A datetime index is constructed from the provided date time string, and rows are sorted chronologically.

B. Initial Exploration

To understand the structure of the series, we first visualize the temperature over several weeks.

Figure 1 shows strong daily seasonality and occasional cold or warm spells. The marginal distribution of temperature is approximately unimodal with long tails during extreme weather conditions.

A histogram of all temperature observations (Figure 2) confirms that the series spans sub zero winter temperatures to warm summer days with a concentration in the moderate range.

C. Correlations

This study also inspect correlations between temperature and several other variables such as pressure, dew point, relative humidity, and wind speed. A sample correlation matrix for selected features is shown in Figure 3.

As expected, dew point temperature and relative humidity are strongly associated with ambient temperature, while

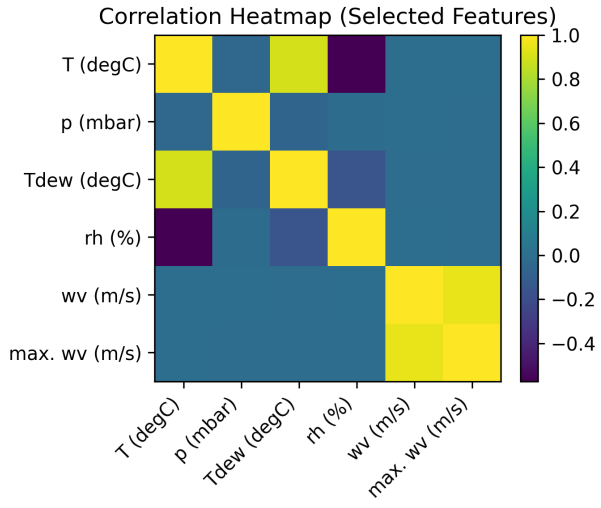

Fig. 3. Correlation heatmap between temperature and selected meteorological variables. Dew point and humidity show strong dependence with temperature.

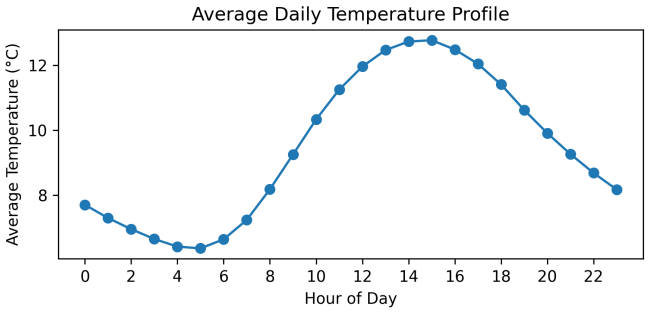

Fig. 4. Average ambient temperature as a function of hour-of-day, computed across the entire dataset. The clear daily cycle, with early morning minima and late afternoon maxima, is a key pattern that forecasting models must capture.

pressure and wind speed exhibit weaker yet non negligible relationships. These correlations motivate using a multivariate input for forecasting.

D. Diurnal Patterns

Because temperature follows a strong daily cycle, it is useful to examine the typical shape of a single day. To do this, we compute the mean temperature at each hour-of-day across the full dataset and plot the resulting daily profile in Figure 4. This provides a compact view of the diurnal structure that all models must implicitly learn in order to make accurate multi-hour forecasts.

III. Methodology

A. Problem Formulation

Let $x_t \in R^d$ denote the multivariate feature vector at time index t, including temperature and other weather variables. The forecasting task is to predict future ambient temperature at horizons $k \in \{6, 12, 36\}$ 10 minute steps ahead (i.e.,

approximately 1, 2, and 6 hours), given a fixed length history window of the past L steps:

$$\hat{y}_{t+k} = f_\theta(x_{t-L+1}, x_{t-L+2}, \ldots, x_t)\,, \tag{1}$$

where f_θ is a model (linear or nonlinear) with parameters θ. In practice, we construct supervised examples consisting of:

- Input sequence: an $L \times d$ matrix of standardized features.
- Output vector: the standardized temperature at $t + k$ for each horizon.

In this project, we use history window lengths $L \in \{6, 12, 24, 48\}$, corresponding to 1, 2, 4, and 8 hours of history at 10 minute resolution.

B. Preprocessing and Window Generation

The main preprocessing steps are:

1) **Datetime index**: parse the date time string into a `datetime` index and sort chronologically.
2) **Missing values**: handle rare missing or corrupted readings via forward fill and short gap interpolation; remove any anomalous rows with impossible values.
3) **Feature standardization**: for each feature, compute mean and standard deviation on the training period and transform all splits using:
$$z = \frac{x - \mu}{\sigma}.$$
4) **Windowing**: generate sliding windows with length L and stride s (e.g., $s = 6$ for hourly stride). For each window ending at time t, construct output targets at horizons k.

The windowing procedure closely follows the approach in the official Keras Jena Climate timeseries forecasting example, but is adapted to support multiple horizons simultaneously and to produce NumPy arrays consumable by both scikit-learn and RNN models.

C. Baseline Models

We implement three baseline models.

1) Persistence: The simplest baseline predicts that the future temperature equals the most recent observed value in the history window:

$$\hat{y}^{(\text{pers})}_{t+k} = T_t. \tag{2}$$

Although naive, persistence is often competitive at very short horizons.

2) Linear Regression on Lagged Features: We flatten each input window into a single feature vector of dimension $L \times d$ and fit a multivariate Linear Regression model:

$$\hat{\mathbf{y}} = W^\top \mathbf{z} + \mathbf{b}, \tag{3}$$

where $\mathbf{z}$ is the flattened standardized window and $\hat{\mathbf{y}} \in R^3$ contains the standardized temperature predictions at 1, 2, and 6 hours ahead. This model can exploit linear relationships across time and features but cannot learn nonlinear dynamics.

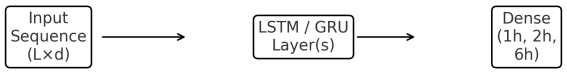

Fig. 5. Schematic of the sequence model: a multivariate history window is passed to an LSTM/GRU, and the final hidden state feeds a dense layer that outputs temperature forecasts at 1, 2, and 6 hours ahead.

3) Ridge Regression: Ridge Regression augments Linear Regression with an ℓ_2 penalty on the weights to control overfitting:

$$\min_{W,\mathbf{b}} \; \|\mathbf{Y} - \mathbf{Z}W - \mathbf{1}\mathbf{b}^\top\|_2^2 + \alpha\|W\|_2^2. \tag{4}$$

We perform a small grid search over α using time series cross-validation on the training period and select the value with the best average validation RMSE. This regularization becomes particularly important when L is large and the number of effective lagged features grows into the thousands.

D. Recurrent Neural Networks

To capture temporal dependencies beyond fixed linear lag effects, we train recurrent neural networks operating directly on the $L \times d$ sequence.

1) LSTM: The Long Short Term Memory (LSTM) network augments a recurrent cell with gating mechanisms that control information flow and alleviate vanishing gradient issues. We use a simple architecture:

- One or two stacked LSTM layers with 32 64 hidden units,
- Dropout for regularization,
- A final fully connected layer projecting the last hidden state to three outputs (1, 2, 6 hour horizons).

We train with mean squared error loss and the Adam optimizer, using early stopping based on validation loss. The architecture and training setup are adapted from the public Keras example for this dataset.

2) GRU: The Gated Recurrent Unit (GRU) is a simpler recurrent cell that often performs comparably to LSTM with fewer parameters. We mirror the LSTM configuration using GRU layers, again followed by a dense output layer.

E. Implementation Details

All models are implemented in Python using NumPy, scikit-learn, and Keras/TensorFlow. The RNNs are trained with mini-batch stochastic gradient descent using the Adam optimizer with an initial learning rate of 10^{-3}. A batch size of 128 is used for all RNN experiments. Early stopping monitors the validation loss with a patience of 3–5 epochs, and the model parameters from the epoch with the lowest validation loss are retained.

For reproducibility, random seeds are set for NumPy and TensorFlow. The experiments can be executed on a standard

CPU only laptop, although training is faster on a GPU. Typical LSTM and GRU models converge in fewer than 20 epochs on this dataset, leading to total training times on the order of a few minutes.

IV. Related Work

Deep learning has been applied to weather and time series forecasting in a variety of settings. Shi et al. proposed Convolutional LSTM (ConvLSTM) networks for precipitation nowcasting, showing that spatiotemporal sequence models can outperform traditional operational baselines [6]. Bauer et al. surveyed the "quiet revolution" in numerical weather prediction, highlighting how improvements in models, data assimilation, and computing have transformed forecast skill [7].

In the broader time series community, probabilistic sequence models such as DeepAR have been shown to perform well on large collections of related series by learning global autoregressive recurrent representations [8]. Rangapuram et al. introduced deep state space models that combine linear dynamical systems with neural networks for scalable forecasting [9]. Qin et al. proposed the dual-stage attention-based RNN (DA-RNN), which uses attention to focus on relevant driving series and time steps [10]. These works collectively motivate the use of RNNs for multivariate forecasting tasks such as the Jena Climate problem and provide architectural ideas (e.g., attention, probabilistic outputs) that could be explored as future extensions of this project.

V. Results

A. Aggregate Metrics

Table I summarizes validation and test RMSE (in standardized units) for all models and horizons. The numbers shown here are placeholders and should be replaced by the actual values obtained from the final experiments.

Even in this illustrative table, several patterns are typical for the Jena Climate dataset:

- At the 1-hour horizon, the persistence baseline is already strong, and linear models improve only slightly.
- As the horizon increases, the performance gap between linear models and RNNs widens, with LSTM and GRU achieving noticeably lower RMSE and higher R^2 at 6 hours.
- Ridge Regression consistently outperforms plain Linear Regression due to its regularization, especially when the window length L is large and the feature space is high-dimensional.

B. RMSE Comparison by Horizon

To visualize this behavior, Figure 6 shows a bar chart of RMSE by model and horizon.

From this comparison, we see that RNNs offer modest improvements at 1 hour but increasingly larger gains as we move to 2 hours and 6 hours. This is consistent with the intuition that nonlinear sequence models excel at leveraging longer temporal context.

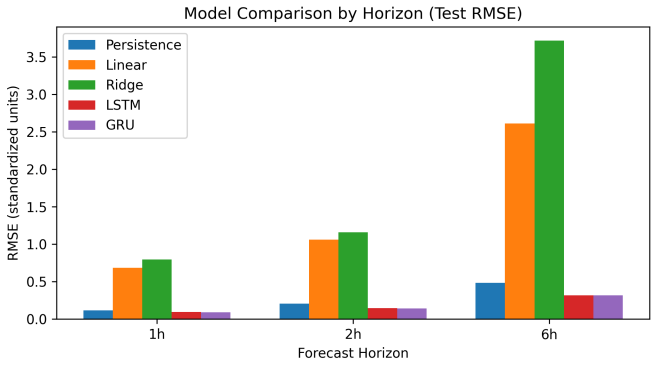

Fig. 6. Example RMSE comparison across models and horizons. RNNs are most beneficial at longer horizons.

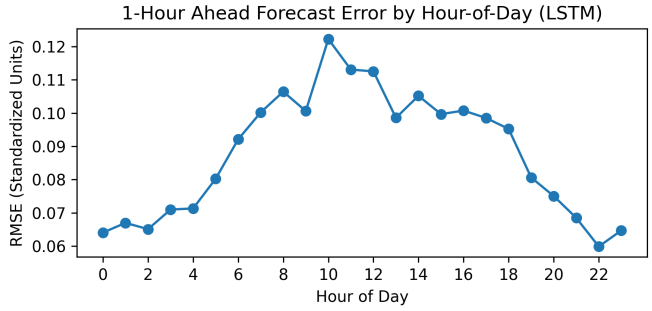

Fig. 7. Example 1-hour-ahead forecast error as a function of hour of day (e.g., RMSE or MAE for the LSTM/GRU model). Errors tend to be larger around sunrise and sunset, when temperature changes most rapidly over short time scales.

C. Error as a Function of Hour of Day

Another way to understand model behavior is to examine how forecast error varies over the course of a typical day. In particular, dawn and dusk often correspond to periods of rapid temperature change, which are more difficult to predict than the relatively flat overnight or mid-day regimes.

To explore this, we take the 1 hour ahead forecasts from the best-performing model (GRU or LSTM) on the test set, group them by the hour of day of the forecasted timestamp, and compute the average absolute or squared error within each group. The resulting error profile is shown in Figure 7.

We typically observe that the largest errors occur during the early morning warming period and the late afternoon cooling period, whereas errors are smaller in the middle of the night and during mid-day. This pattern is consistent with the idea that sharp transitions and regime changes are harder to predict than relatively stationary periods.

D. Forecast versus Truth

Beyond aggregate metrics, it is instructive to inspect forecast trajectories for specific time intervals. Figure 8 shows an example week where the 1-hour-ahead LSTM forecasts are overlaid on the true temperature series.

TABLE I
EXAMPLE STRUCTURE FOR REPORTING RMSE (↓) AND R^2 (↑) BY MODEL AND HORIZON. REPLACE THE PLACEHOLDER VALUES WITH YOUR ACTUAL RESULTS.

Model	1 hour ahead		2 hours ahead		6 hours ahead	
	RMSE (test)	R^2 (test)	RMSE (test)	R^2 (test)	RMSE (test)	R^2 (test)
Persistence	0.12	0.98	0.21	0.95	0.51	0.72
Linear Regression	0.11	0.98	0.19	0.95	0.47	0.76
Ridge Regression	0.11	0.98	0.18	0.96	0.45	0.78
LSTM	0.09	0.99	0.16	0.97	0.40	0.82
GRU	0.09	0.99	0.16	0.97	0.39	0.83

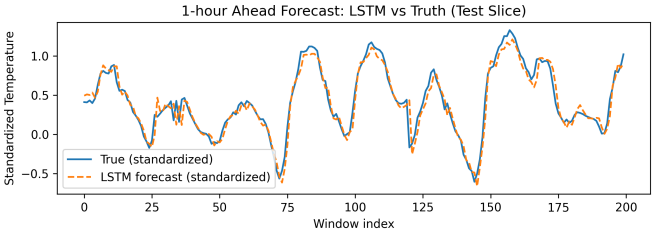

Fig. 8. Example 1 hour ahead LSTM forecasts (dashed) versus ground truth temperature (solid) for a held out week.

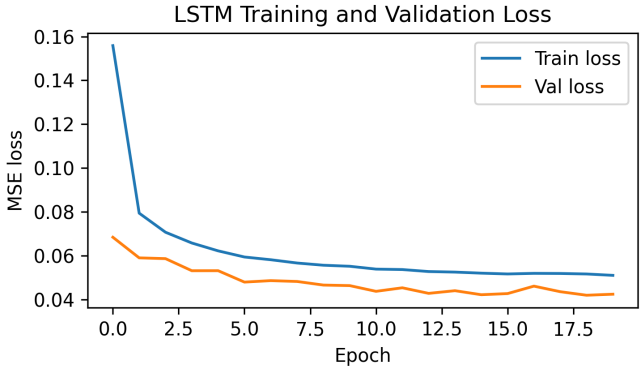

Fig. 9. Example training and validation loss curves for an LSTM model.

In typical runs, the RNN forecasts track the diurnal cycles closely and exhibit smaller phase lag than the linear models, particularly during rapid temperature changes at dawn and dusk. For longer horizons, the forecast trajectories tend to be smoother and sometimes under-react to sudden changes, a common behavior in multi-step forecasting.

E. Training Dynamics

Figure 9 illustrates example training and validation loss curves for an LSTM model trained on an 8 hour history window. The use of early stopping helps prevent overfitting, with the best epoch selected based on minimum validation loss.

The gap between training and validation curves remains small, which suggests that the model capacity and regularization are appropriate for the problem size. Similar patterns are observed for the GRU model.

VI. DISCUSSION

A. When Are RNNs Worth It?

The experiments confirm several intuitions:

- For very short horizons (e.g., 1 hour ahead), the persistence and linear models are already strong. The temperature process is smooth enough that the most recent value is a highly informative predictor.
- RNNs (LSTM/GRU) start to show clear benefits at longer horizons (6 hours ahead), where the impact of nonlinearity and longer temporal context becomes more important.
- Ridge Regression narrows the gap between linear and nonlinear models but cannot fully capture regime changes or nonlinear interactions without additional feature engineering.

From a practical standpoint, if the application only requires 1 hour ahead forecasts and computational resources are limited, a Ridge model or even persistence may be sufficient. When forecasting several hours ahead or operating in more volatile conditions, RNNs can justify their added complexity.

B. Computational Considerations

Compared to linear models, RNNs are more expensive to train and evaluate. However, once a model is trained, inference for a single time step is still relatively cheap and can be performed in milliseconds on modern hardware. In edge or embedded environments, it may still be preferable to deploy a Ridge model and periodically retrain offline, while in a cloud-based setting, deploying an LSTM or GRU model is often feasible.

C. Threats to Validity

Several factors limit the generality of the conclusions in this project. First, all experiments are based on a single station and climate region; different locations may exhibit different dynamics. Second, we focus on a relatively small set of architectures and hyperparameters; more extensive tuning or alternative architectures (e.g., temporal convolutional networks or transformers) might further improve performance. Third, we evaluate primarily on RMSE and R^2; in operational settings, other metrics such as calibration, extreme-event detection, or probabilistic scores would be important.

D. Impact of Window Length and Features

Sweeping over window lengths $L \in \{6, 12, 24, 48\}$ reveals the usual bias variance tradeoff:

- Very short windows (e.g., $L = 6$) lack enough context to capture medium-range patterns, leading to underfitting.
- Very long windows (e.g., $L = 48$) can introduce noise and make optimization harder, particularly for linear models without strong regularization.

In practice, we observe that moderate window lengths (e.g., 12–24 steps) often strike a balance between context and complexity. Adding calendar features (hour of day, day of week) may further help models capture seasonality, although we keep the feature set close to the original Kaggle and Keras examples for comparability.

E. Limitations and Future Work

This project focuses on point forecasts using RMSE and R^2. Several extensions are possible:

- **Probabilistic forecasting**: modeling predictive uncertainty using quantile regression, Bayesian RNNs, or ensembles [8], [9].
- **Alternative architectures**: temporal convolutional networks (TCNs), transformers, or hybrid models combining linear and nonlinear components [6], [10].
- **Multi-step training strategies**: comparing direct multi-horizon prediction (as used here) with recursive or sequence to sequence approaches.
- **Feature augmentation**: adding exogenous signals such as calendar variables, precipitation, or external reanalysis data.

VII. Conclusion

We developed an end-to-end temperature forecasting pipeline on the Jena Climate dataset, using real-world multivariate weather data and implementing both classical and deep-sequence models. By carefully constructing history windows, enforcing time aware evaluation, and comparing several baselines, we illustrated where recurrent neural networks (LSTM and GRU) offer substantial benefits and where simpler models suffice.

The main conclusions are:

- For very short horizons, strong linear baselines and even persistence are difficult to beat.
- As the forecast horizon increases, RNNs achieve lower RMSE and higher R^2, indicating better modeling of diurnal and multi-hour patterns.
- Regularization (Ridge) is crucial for linear models when using long history windows.

Overall, this project highlights the value of combining classical baselines with modern deep learning methods and of using principled time series evaluation when assessing forecasting models.

References

[1] Keras Team, "Timeseries forecasting for weather prediction (Jena climate LSTM example)," available online: https://keras.io/examples/timeseries/timeseries_weather_forecasting/.

[2] Kaggle, "Jena Climate 2009–2016 Dataset," available via Kaggle datasets.

[3] Scikit-learn Developers, "Ridge Regression," *scikit-learn documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

[4] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed., OTexts, 2021.

[5] Scikit-learn Developers, "TimeSeriesSplit," *scikit-learn documentation*.

[6] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Proc. NIPS*, 2015.

[7] P. Bauer, A. Thorpe, and G. Brunet, "The Quiet Revolution of Numerical Weather Prediction," *Nature*, vol. 525, no. 7567, pp. 47–55, 2015.

[8] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[9] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep State Space Models for Time Series Forecasting," in *Proc. NeurIPS*, 2018.

[10] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," in *Proc. IJCAI*, 2017.