Julen Legido 268309 Alfonso Castelijns 254267 Adrià Porta 268513

Project Report

Introduction

Weather forecasting plays a vital role in numerous sectors, from agriculture and infrastructure planning to disaster management and public safety. With the increasing availability of historical weather data, machine learning techniques have become valuable tools for improving the accuracy of short- and long-term weather predictions.

In this project, we implement a Long Short-Term Memory (LSTM) neural network to forecast daily temperature based on multiple historical weather variables. Our approach is inspired by the methodology presented in a Medium article by Ozdogar[1], which demonstrates the effectiveness of LSTMs for time-series weather forecasting tasks.

We develop our model using PyTorch, and structure it to learn from multivariate time-series data, where the network uses several weather variables over the past 31 days to predict the temperature for the following day.

State of the Art

Over the past decade, a wealth of research has emerged exploring how machine learning and deep learning models can be cleverly adapted to improve the accuracy and explainability of weather predictions.

Our project was inspired by the implementation described in the Medium article *“Time Series Forecasting Using LSTM – PyTorch Implementation”* by Ozdogar ([1]). In that article, the author presents a clean and practical approach to time series prediction using LSTMs with PyTorch. The model predicts future values based on a sequence of past observations, and the article highlights key steps including data scaling, sequence creation, and model architecture. We have adapted this framework to a multivariate weather forecasting context, enhancing it with additional baseline models and explainability techniques to broaden its applicability and interpretability.

Long Short-Term Memory networks are a brilliant innovation in the realm of recurrent neural networks, were specifically designed to capture long-range temporal dependencies. Unlike traditional Recurrent Neural Networks (RNNs), LSTMs are equipped with memory cells and gating mechanisms that help them selectively remember or forget information over extended sequences. This makes them particularly well-suited for weather forecasting tasks, where past weather patterns often influence future conditions in complex and subtle ways. Numerous recent studies have demonstrated the superior capability of LSTMs in time-series prediction tasks ranging from stock prices to rainfall estimation, which reinforces our motivation to explore them in depth.

Classic RNNs are an earlier form of sequence modeling that process data step-by-step, maintaining a hidden state across time. However, due to their simpler architecture, they often suffer from the vanishing gradient problem, which hinders their ability to learn long-term patterns. Nevertheless, they provide a valuable baseline and can still perform reasonably well on shorter sequences. In our project, we include a traditional RNN as a comparative benchmark to see how much performance gain LSTMs offer in our specific weather prediction scenario.

Gaussian Processes (GPR) is a non-parametric, Bayesian approach to regression that models uncertainty in a very principled and elegant way. It provides not just a prediction, but also a confidence interval, which is particularly useful in critical applications like weather forecasting. Although GPR struggles to scale with larger datasets, its interpretability and reliability make it a powerful baseline, especially when the data is moderately sized and the goal includes understanding model confidence.

XGBoost (Extreme Gradient Boosting) is an incredibly powerful and widely adopted ensemble learning technique known for its exceptional performance in structured data tasks. It constructs a series of decision trees in a stage-wise manner and optimizes for both speed and accuracy. XGBoost doesn't natively handle time-series data, but with careful feature engineering, it can still be an effective comparator, especially when combined with lag-based inputs. Its robustness and efficiency make it a standard benchmark in many machine learning competitions.

SHapley Additive exPlanations (SHAP) is a rapidly growing and vitally important field in AI. SHAP leverages game theory to provide fair and consistent attributions of feature importance. It assigns each feature an importance value for a particular prediction by computing Shapley values, which fairly distribute the "credit" among input features. SHAP has become one of the most trusted tools for interpreting black-box models, offering deep insights into why models make the predictions they do. Integrating SHAP into our project allows us to move beyond accuracy metrics and explore how and why our model reaches its conclusions—an essential step toward transparency and trust in AI.

Methodology

Data Analysis

In this project, we used a historical weather dataset of New York City, spanning from January 2023 to December 2024. The data was obtained from Visual Crossing Weather and contains daily weather observations temperature, humidity, wind speed, visibility, pressure, etc.

To prepare the dataset for modeling, several preprocessing steps were undertaken to ensure the data was clean, consistent, and compatible with machine learning algorithms. First, we removed irrelevant or redundant columns that did not contribute meaningful information for prediction.

Next, we handled categorical data by converting relevant features into numerical format. The 'preciptype' and 'conditions' columns were encoded using categorical codes. These encodings enabled the use of categorical variables in numerical models, and the mappings were stored to allow for later interpretation, particularly in explainability analyses using SHAP.

Additionally, we transformed the time-based features 'sunrise' and 'sunset'. These columns, originally in string format, were first converted to Python datetime objects and then transformed into the number of minutes after midnight. This numeric representation made them more suitable as inputs to machine learning models. The original datetime columns were then dropped from the dataset.

To ensure stability during neural network training, all numerical features were scaled to a common range using MinMaxScaler. This scaling step is especially important for gradient-based models such as LSTM and RNN, as it improves convergence and model performance.

To properly evaluate the model's generalization capabilities, we split the dataset into 80% training and 20% testing sets. As this is a time series forecasting task, we avoided randomly shuffling the data. Instead, we applied a sequential split, using the first 18 months (January 2023 to June 2024) for training and the final 6 months for testing.

Models and Optimization

Our LSTM model was designed to learn complex temporal patterns from multiple input features. We used a sliding window of 31 days, with each input sequence capturing the multivariate weather conditions leading up to the prediction day. The model architecture consisted of an input size of 25 features, a hidden size of 64 units, 2 LSTM layers and as output layer a fully connected layer mapping the last hidden state to a single temperature prediction.

The model was trained using the Adam optimizer with a learning rate of 0.001, and Mean Squared Error (MSE) was used as the primary loss function. Mean Absolute Error (MAE) was also tracked during training. Training was conducted over 100 epochs with a batch size of 32.

To evaluate the effectiveness of the LSTM, we also trained some baseline models on the same data (flattened to 2D for compatibility).

For the Gaussian Process Regressor we used a composite kernel (RBF × Constant Kernel) with 10 restarts for hyperparameter optimization. GPR performed relatively well, capturing nonlinear relationships but suffering from scalability constraints.

For the Extreme Gradient Boosting we trained an XGBoost model with 100 estimators, a maximum depth of 7, and a learning rate of 0.1. XGBoost delivered strong predictive performance with fast training and interpretability through built-in feature importance.

As a baseline neural architecture, we implemented a shallow RNN with one recurrent layer (hidden size = 32) followed by a fully connected output layer. Though computationally lighter, the RNN struggled to capture long-term dependencies compared to LSTM.

All models were evaluated using MAE, MSE and R^2 Score. These metrics provided insight into absolute accuracy, variance in prediction, and overall goodness-of-fit, respectively.

To interpret model behavior and feature importance, we applied SHAP analysis to the XGBoost model, which is inherently compatible with SHAP's tree explainer. Feature importance was analyzed across the 31-day flattened input space, with each time-lagged feature labeled as feature_t-n.

The SHAP summary plot provided global insights into which lagged weather features most influenced predictions. Additionally, a waterfall plot was generated to visualize the contribution of individual features for a specific prediction. This helped validate that the model relied on meaningful and interpretable patterns in the data.

Experiments

To assess the effectiveness of our proposed LSTM model and compare it against several models that were used as baselines.

All models were trained on the same processed dataset, using the first 80% of the time series for training and the last 20% for testing, preserving the chronological order to reflect realistic forecasting conditions. Each data sample consisted of a 31-day sliding window of weather variables as input, predicting the average temperature of the following day.

The LSTM model was trained using the Adam optimizer with a learning rate of 0.001, a value recognized generally as the default value (https://www.geeksforgeeks.org/adam-optimizer/). Since Adam dynamically adjusts learning rates for each parameter during training, starting with 0.001 provides a good balance between divergence and training speed. We monitored both MSE as the primary loss function and MAE to track absolute deviations during training. Training was performed for 100 epochs with a batch size of 32.

To establish meaningful baselines, we also trained a Simple RNN, a GPR, and an XGBoost model. The Simple RNN had a reduced architecture with one recurrent layer of 32 hidden units. While it is computationally lighter than LSTM, it lacks memory gating mechanisms, making it less effective at modeling long-term dependencies. The GPR model employed a composite kernel and was optimized with 10 restarts; although it offered probabilistic predictions, its scalability was limited due to the growing computational cost with larger datasets. The XGBoost model was trained using 100 estimators, a maximum depth of 7, and a learning rate of 0.1.

All models were evaluated using the same three metrics: MAE, MSE, and R^2 score. In addition to numerical evaluation, we visualized model predictions against actual values over the full timeline to assess how well each model captured variations.

SHAP explainability analysis was applied to the XGBoost model to quantify the impact of individual time-lagged features on the output prediction. This analysis revealed which specific features were most impactful in our predictions.

These experiments were carefully designed to test specific hypotheses such as whether sequence models outperform non-sequential baselines on this forecasting task, how model complexity influences generalization and which features play a dominant role in predicting temperature.

Results

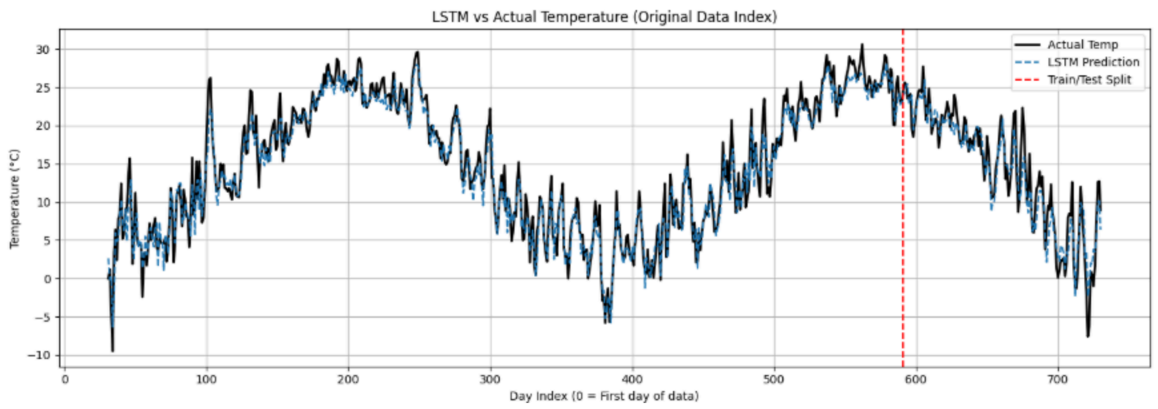
The LSTM model achieved the best overall performance, with an average R² score of 0.8955, an MAE of 1.973, and an MSE of 6.4115. These results demonstrate the model's strong capacity to capture temporal dependencies and generalize well to unseen data. The Simple RNN also performed well, achieving an R² score of 0.8896, an MAE of 1.9872, and an MSE of 6.7737, though it slightly underperformed compared to the LSTM there were iterations where it outperformed it that we attribute to randomness of sampling and training due to our smaller dataset and not that deep models.

Among the non-sequential models, XGBoost delivered competitive results with an R² score of 0.8650, an MAE of 2.2417, and an MSE of 8.2851. Although it does not inherently model time dependencies, its ability to capture complex nonlinear feature interactions enabled it to perform effectively when fed with flattened temporal inputs. The Gaussian Process Regressor (GPR) showed the weakest predictive performance, with an R² score of 0.8253, an MAE of 2.6328, and an MSE of 10.7245. This result reflects the model's limited scalability and reduced suitability for high-dimensional input sequences.

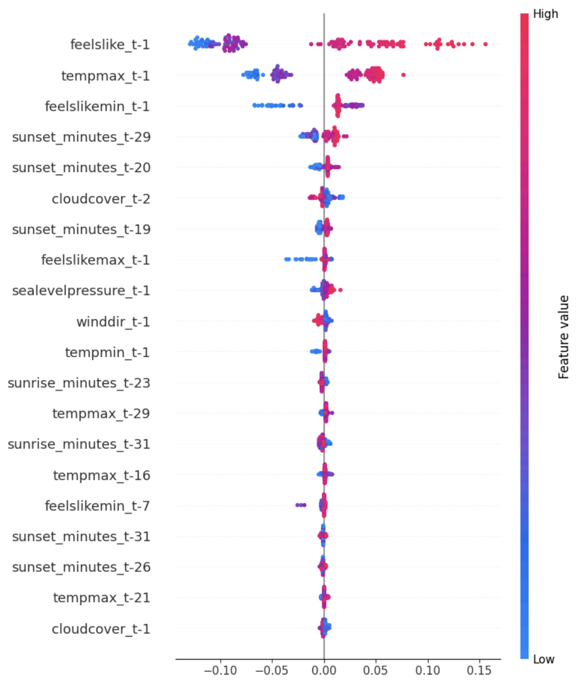
It is important to note that due to the design of our experiment, the non-sequential models (XGBoost and GPR) were always trained on the same flattened representations of the input data, without any variation in the sequential structure across runs. As a result, they consistently produced the same outputs, not giving us the best representation of their capabilities.

This design constraint limited the non-sequential models' ability to benefit from the natural variability and temporal dynamics present in time-series data. In contrast, the sequential models (LSTM and RNN) were able to utilize the full structure of the 31-day input sequences, leading to more adaptive and temporally aware predictions.

Overall, the LSTM model delivered the most accurate results, followed closely by the RNN. While the non-sequential models performed well given their constraints, their inability to model time directly was a limiting factor in capturing complex temporal trends. Below we can see the actual temperatures compared to the predicted ones by the LSTM.



In terms of SHAP, the results revealed that the model relies most heavily on recent historical observations, especially temperature-related variables. In particular, features such as feelslike_t-1, tempmax_t-1 and feelslikemin_t-1 (as we can see in the graph below) were consistently among the top contributors to the predictions. This indicates that the most recent days' maximum, minimum, and perceived temperatures are critical in guiding the model's understanding of future temperature trends.



Discussion & Future work

Our experiments demonstrate that sequence models—particularly the LSTM—outperform both classical baselines and simpler RNNs when forecasting daily temperatures. The LSTM's gating mechanisms allow it to capture both short-term fluctuations and longer seasonal trends, yielding the highest R^2 (0.8955) and lowest MAE and MSE. The simple RNN achieved nearly comparable accuracy ($R^2 = 0.8896$), suggesting that even minimal recurrence provides substantial benefit over non-sequential methods. In contrast, XGBoost ($R^2 = 0.8650$) and Gaussian Processes ($R^2 = 0.8253$) showed respectable performance but were constrained by their inability to natively model temporal dependencies in a sliding-window context. SHAP analysis further revealed that the most recent day's temperature and "feels-like" metrics dominate the model's decision process, confirming that immediate history is the strongest driver of next-day temperature.

Despite these strengths, our study has limitations. First, we fixed the sequence length at 31 days and did not explore shorter or longer windows systematically. Second, we relied solely on surface-level weather variables; integrating auxiliary data—such as atmospheric pressure maps, satellite imagery, or large-scale climate indices—could enhance long-range forecasting. Third, our hyperparameter tuning was limited (e.g., fixed hidden sizes and learning rates); a more exhaustive search or automated optimization (e.g., Bayesian search) might yield further gains. Finally, while SHAP provided useful global and local explanations, it remains computationally intensive for deep models; alternative explainability methods (e.g., attention mechanisms within the network) could offer more efficient interpretability.

Future work will therefore focus on:

Window and architecture exploration: systematically varying sequence lengths, adding attention layers, or experimenting with Transformer-style encoders for improved long-term pattern capture.

Feature enrichment: incorporating external meteorological datasets (e.g., regional pressure fields, humidity profiles aloft) and engineered seasonal indicators to improve predictive context.

Broadening the geographic scope: this work could include other cities —such as Valencia, Spain— to reframe the task from temperature forecasting to anomaly detection, particularly focusing on extreme rainfall events. By identifying patterns that precede unusually high precipitation, the model could serve as an early warning tool for flood risk management. Such an application would have significant societal value, especially in regions increasingly affected by climate change and urban flooding.

Automated tuning: applying hyperparameter optimization frameworks to jointly tune network depth, learning schedules, and regularization to guard against overfitting.

Deployment and robustness: evaluating model performance in an online setting with continuous data ingestion, and extending uncertainty quantification beyond SHAP (e.g., Monte Carlo dropout) to provide actionable confidence bands for decision-makers.

Conclusions

In summary, our multivariate LSTM achieved the best balance of accuracy and temporal awareness when forecasting next-day temperatures, validating our hypothesis that gated recurrence effectively models weather time-series. The simple RNN baseline confirmed the value of even basic memory, while XGBoost and Gaussian Processes offered useful non-sequential perspectives with competitive—but ultimately lesser—performance. Explainability via SHAP underscored the primacy of recent temperature metrics in driving predictions. Moving forward, expanding our feature set, refining architectures, and automating optimization promise further improvements, bringing us closer to a robust, explainable weather-forecasting system.

References

1. Özdogar, A. (2023, September 9). *Time series forecasting using LSTM – PyTorch implementation*. https://medium.com/@ozdogar/time-series-forecasting-using-lstm-pytorch-implementation-86169d74942e