**FORECASTING TEMPERATURES USING TIME SERIES TECHNIQUES**

**A CAPSTONE PROJECT REPORT**

(DATS 6501, SECTION 11)

**BY**

Hrushikesh Uppalapati (G44588465)

Chekitha Swayampu (G29426782)

**Under the guidance of**

Dr. Abdi Awl



April 30, 2025

**ABSTRACT**

Accurate temperature forecasts help farmers safeguard crops, utilities balance supply and demand, and city planners plan for harsh weather. In this project, we converted approximately three million daily temperature observations from US cities (1995-2019) into user-friendly forecasts using a simple, end-to-end procedure.

First, we cleaned the data by consolidating separate year, month, and day columns into a single date field, retaining only U.S. records, removing duplicates, filling in small gaps, and filtering out obvious outliers. Then we added "memory" to our models, including lagged temps (yesterday, last week, last month), rolling averages (7- and 30-day), simple season tags, calendar details, and city/state codes.

We tried five different methods: ARIMA and its seasonal relative, SARIMA for classic time-series trends, Random Forest and XGBoost for powerful ensembles that capture complicated patterns, and LSTM neural networks for learning long-term sequences. We automatically tweaked each strategy and measured how close their predictions came to reality (using RMSE, MAE, and R² values). XGBoost led the pack with an RMSE of 3.74°F, followed by Random Forest and LSTM, while ARIMA and SARIMA offered clear baselines.

To make these findings more visible, we created two interactive dashboards: one allows you to overlay forecasts and actuals for any city and model, while the other maps out error metrics across space and season. Although our models do not currently incorporate humidity or wind, and SARIMA only works at the monthly level, our open-source framework provides robust, customizable forecasts. The next steps include adding more weather variables, investigating probabilistic techniques, and providing real-time updates to keep predictions current.

**TABLE OF CONTENTS**

Abstract ..............................................................................................................................1

Glossary of Terms ..............................................................................................................4

1. Introduction ......................................................................................................................6 1.1 Introduction and Background ................................................................................. 6 1.2 Problem Statement .................................................................................................. 6 1.3 Problem Elaboration ................................................................................................. 7 1.4 Motivation ................................................................................................................ 7 1.5 Project Scope ............................................................................................................... 8
2. Literature Review ..........................................................................................................8 2.1 Relevant Research ........................................................................................................ 8
3. Methodology ................................................................................................................ 10 3.1 Dataset Description ................................................................................................ 10 3.2 Data Collection ...................................................................................................... 10 3.3 Data Preprocessing ................................................................................................ 10 3.4 Feature Engineering ............................................................................................... 12 3.5 Data Modeling ....................................................................................................... 14 3.5.1 ARIMA ............................................................................................................... 14 3.5.2 Random Forest .................................................................................................... 16 3.5.3 XGBoost .............................................................................................................. 18 3.5.4 LSTM ................................................................................................................... 20 3.5.5 SARIMA .............................................................................................................. 22 3.6 Visualizations .......................................................................................................... 25 3.6.1 Forecast Comparison Dashboard ......................................................................... 25 3.6.2 Error Analysis Dashboard ....................................................................................... 26
4. Results and Analysis ..................................................................................................... 28 4.1 Model Performance Comparison ............................................................................ 28 4.2 City-Level Forecast Error Variation ........................................................................... 29
5. Conclusion ...................................................................................................................... 30 5.1 Conclusion ................................................................................................................ 30 5.2 Project Limitations ................................................................................................... 31 5.3 Future Research ......................................................................................................... 31
6. References ........................................................................................................................ 32
7. Appendix ...................................................................................................................33 Appendix A1: Lag and Rolling Average Features ......................................................... 33 Appendix A2: Feature Correlation and Temporal Patterns .............................................. 3

**GLOSSARY OF TERMS**

1. **ARIMA (Autoregressive Integrated Moving Average)**: A statistical model for time series forecasting that combines autoregression (using past values), integration (differencing to remove trends), and moving averages (smoothing forecast errors).
2. **SARIMA (Seasonal ARIMA)**: An extension of ARIMA that adds seasonal autoregressive, differencing, and moving-average terms to model repeating cycles (e.g., annual temperature patterns).
3. **LSTM (Long Short-Term Memory)**: A recurrent neural network with gated memory cells, designed to learn long-range dependencies in sequential data such as daily temperature series.
4. **Random Forest**: An ensemble machine learning method that builds multiple decision trees on random feature subsets and averages their outputs, improving accuracy and reducing overfitting.
5. **XGBoost (eXtreme Gradient Boosting)**: It is a gradient-boosted decision tree technique that constructs trees progressively, with each tree rectifying the flaws of its predecessors, and is improved by regularization and fast, parallelized training.
6. **Lag Feature**: A predictor constructed by shifting the target variable by a defined number of time steps (for example, the prior day's temperature) to capture autocorrelation in time series.
7. **Rolling average**: Is a smoothed value computed by averaging a fixed-size window of consecutive observations (e.g., 7-day or 30-day mean) to emphasize long-term trends over short-term variations.
8. **IQR (Interquartile Range)**: It is a statistical dispersion measure equal to the difference between the 75th (Q3) and 25th (Q1) percentiles. It identifies and eliminates outliers beyond [Q1 - 1.5×IQR, Q3 + 1.5×IQR].
9. **Outlier**: An observation that deviates significantly from other values, usually due to measurement error or infrequent events; removing outliers can enhance model stability.
10. **RMSE (Root Mean Square Error):** It is a forecast accuracy statistic that is derived as the square root of the average squared discrepancies between predictions and actual values. It is sensitive to big errors.
11. **MAE (Mean Absolute Error)**: It is the average of the absolute differences between predicted and actual values, which provides a simple measure of average forecast deviation.
12. **R² (Coefficient of Determination)**: It is a statistical measure that indicates how well anticipated values explain observed data variation. It ranges from 0 (no explanatory power) to 1 (perfect fit).
13. **One-Hot Encoding:** is a technique for converting categorical data into binary columns (one per category), allowing algorithms to handle non-numerical properties without suggesting ordinal relationships.
14. **Label Encoding:** is a method of mapping each category in a categorical variable to a unique integer that preserves category distinctions while lowering dimensionality when compared to one-hot encoding.
15. **STL (Seasonal-Trend Decomposition Using Loess):** It is a decomposition approach that divides a time series into trend, seasonal, and residual components using locally weighted regression (LOESS).
16. **MSTL (Multiple Seasonal-Trend Loess)**: An extension of STL capable of decomposing multiple overlapping seasonal patterns (e.g., daily, weekly, yearly) in high-frequency time series.
17. **INTRODUCTION**

**1.1 INTRODUCTION AND BACKGROUND**

Every day, temperature changes affect our society: farmers rely on accurate forecasts to protect crops from unexpected frosts; energy providers modify production to meet heating or cooling demands; and city planners anticipate heat waves to protect sensitive populations. Despite its importance, predicting temperature remains difficult due to complicated seasonal fluctuations, unexpected weather events, and long-term climate trends. While useful for capturing broad trends, traditional statistical methods frequently fail to adapt when underlying conditions change or data shows nonlinear relationships.

In recent years, the combination of machine learning with time series analysis has opened up new opportunities. Models can learn both rapid variations and repeating cycles in temperature data using engineering features such as rolling averages, lagged observations, and seasonal markers. Deep learning architectures such as LSTM improve this skill by remembering patterns over longer periods, but ensemble methods such as Random Forest and XGBoost excel at managing noisy datasets with multiple interdependent inputs.

This capstone project builds on previous accomplishments to develop a strong, end-to-end forecasting pipeline. Beginning with nearly three million daily temperature records taken from reliable repositories, we thoroughly clean and preprocess the data, removing outliers, filling gaps, and embedding temporal and geographic features. Our mission is straightforward: use data-driven insights to provide trustworthy, actionable temperature predictions that enable better decision-making across multiple industries.

**1.2 PROBLEM STATEMENT**

Temperature forecasting is critical for businesses such as agriculture and energy planning. Traditional approaches frequently lack accuracy owing to complicated data patterns. This project employs modern data science techniques such as time series analysis and machine learning on a large set of data from multiple countries and states, totaling approximately 2.9 million records, to provide precise daily and weekly forecasts. These insights are intended to help guide decision-making, decrease risks, and improve resource planning.

**1.3 PROBLEM ELABORATION**

Accurate temperature forecasts are crucial in agriculture, energy management, and urban planning, but even small inaccuracies can have serious consequences. Missed frost alerts can destroy crops, and underestimating peak demand during heat waves increases the danger of power outages. Traditional time-series techniques, such as ARIMA, excel at modeling linear trends and fixed seasonal cycles, but they struggle to describe nonlinear temperature dynamics or unexpected anomalies, such as sudden cold snaps or heat-island effects. Our dataset exacerbates the problem: approximately three million daily records from dozens of U.S. cities introduce various local climates and need individualized models, increasing computational costs for parameter adjustment and model training. Deep learning approaches like LSTM have the potential to capture long-term dependencies, but they require careful sequence design, feature scaling, and significant computational resources. Furthermore, stakeholders require clear indicators of uncertainty in addition to point estimates, which is why we emphasize MAE and RMSE. To successfully address this issue, a coherent pipeline is required that cleans and imputes missing values, removes outliers, incorporates lag, rolling-average, and seasonal characteristics, and balances model accuracy and runtime. By doing so, we hope to provide reliable, context-aware temperature forecasts that assist fast, data-driven decision-making across industries.

**1.4 MOTIVATION**

Every day, temperature projections inform decisions about our food, energy, and cities. Farmers rely on accurate forecasts to safeguard sensitive crops from sudden frosts, and power companies balance generation and demand to keep the lights on during heatwaves. Weather forecasts help city planners plan for extreme occurrences and protect vulnerable neighborhoods. However, classic forecasting approaches frequently miss rapid swings or fail to adjust when local climatic patterns change.

This project is motivated by the real-world demand for more precise, context-aware temperature predictions. We hope to capture repeating seasonal cycles as well as unanticipated anomalies by integrating statistical models with modern time series, machine learning, and deep learning approaches. Our mission is to transform raw data into clear, actionable insights, allowing growers, utilities, and industries to make more informed, quicker decisions, decrease risks, and allocate resources more efficiently.

**1.5 SCOPE**

The goal of this research is to put together an end-to-end temperature forecasting system for US cities using a large-scale daily dataset from the University of Dayton (Kaggle). We conducted our investigation on historical temperature data to assess the core effectiveness of time series and machine learning techniques. Data cleansing (missing value imputation, outlier removal), feature engineering (lagged values, rolling averages, seasonal and calendar indicators, geographic encodings), and model training with ARIMA, SARIMA, Random Forest, XGBoost, and LSTM are important components. Models are compared by MAE, RMSE, and R² measures. We also provide interactive dashboards for visualizing forecast trends and spatial error distributions. This project does not include real-time streaming forecasts or the integration of external meteorological sources; those are for future work. A written report, replicable codebase, and dashboard prototypes will be delivered to help data-driven decision-making in agricultural, energy, and urban planning.

1. **LITERATURE REVIEW**

This literature review looks at recent developments in temperature forecasting, beginning with classical time-series models (ARIMA, SARIMA) used on regional datasets and global anomaly investigations. We then investigate machine learning ensembles such as Random Forest and XGBoost, hybrid frameworks that combine statistical and neural methods, and multi-seasonal decomposition techniques. This research informs the architecture of our forecasting pipeline, which spans multiple climates and timescales.

**2.1 RELEVANT RESEARCH**

Weather forecasting continues to rely heavily on time series methods. Ashraf (2024) assessed ARIMA and SARIMA at three UK weather stations, selecting the best p, d, and q parameters for each site using ACF/PACF and formal stationarity tests (ADF). The study found that seasonal ARIMA models outperformed non-seasonal models in accurately capturing annual cycles, emphasizing the significance of station-specific parameter tweaking. Other researchers conducted a similar multi-station analysis on UK station data, comparing classical methods such as Holt-Winters exponential smoothing, and found that models customized to local seasonal patterns greatly cut forecast error.

Machine learning techniques have supplemented these statistical models. Rajasekaran Meenal et al. (2021) used Random Forest to estimate sun radiation and wind speed, with training on lag, calendar, and geographical variables. The model achieved an R² of 0.97 and an MSE of 0.75, indicating that ensemble tree approaches may capture nonlinear interactions in meteorological data with limited explanatory factors.

Hybrid pipelines that mix linear, statistical, and ensemble approaches have further benefits. Singh et al. (2024) examined linear regression, ARIMA, SARIMA, and ARIMAX on regional meteorological data, and then ensembled ARIMAX and SARIMA forecasts. This ensemble lowered MSE to 0.223, much below individual model errors, and demonstrated the value of integrating complementary techniques.

Long-term temperature anomalies can be analyzed to assist in creating more accurate forecasting models. Kumar et al. (2020) used NOAA's 1880–2017 global anomaly dataset, 137 years of monthly departures from a baseline, to fit a seasonal ARIMA(1,1,1)(1,1,0,12) model. They demonstrated that anomaly-based systems capture gradual climatic trends well, but cautioned that translating projections back into actual temperature readings adds significant complexity and demands rigorous conversion methods to preserve accuracy and provide insights for climate study.

Nguyen et al. (2022) created a compact, end-to-end forecasting framework customized to Ho Chi Minh City's microclimate. They collected hourly and daily temperature data and tested a variety of models, classical ARIMA, exponential smoothing, pure machine learning (e.g., Random Forest), and hybrid combinations of neural networks and regression trees across various historical window lengths. Using MAE and RMSE as benchmarks, their hybrid ARIMA-ANN technique produced the highest short-term accuracy. The system, implemented in Python, has modular data ingestion, automated model selection, and dashboard visualizations, making it particularly ideal for smart-agriculture indoor situations where large-scale forecasting platforms are difficult.

Finally, developments in decomposition provide additional information for forecasting. Bandara, Hyndman, and Bergmeir's (2021) MSTL algorithm extends STL to accommodate several seasonal cycles using iterative LOESS smoothing. MSTL efficiently separates trend and each seasonal component daily, weekly, and yearly at a low computational cost. It is provided in the R forecast package and provides a powerful preprocessing tool for high-frequency, multi-seasonal datasets.

1. **METHODOLOGY**

**3.1 DATASET DESCRIPTION**

The “Daily Temperature of Major Cities” dataset from Kaggle contains daily average temperature readings for roughly 400 global cities between January 1, 1995, and December 31, 2019, totaling approximately 2.9 million records. Each row includes the city's region, country, state (if relevant), name, calendar fields (year, month, and day), and average temperature in Fahrenheit. The CSV (about 350 MB) is uniformly formatted and has consistent date coverage. Missing items (approximately 0.2%) are due to reporting gaps and are indicated for imputation. Its breadth makes it excellent for assessing statistical and machine learning-based forecasting methodologies on a large scale.

**3.2 DATA COLLECTION**

We retrieved the raw CSV file directly from the Kaggle repository, [Daily Temperature of Major Cities](https://www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-major-cities). We used Python's pandas package to load the file with date parsing enabled, which combined year, month, and day into a unified datetime index and cast region, country, state, and city as categorical categories. Next, we filtered the data to include only records with the country "United States," which included all of the original set's U.S. states and cities. No external datasets were combined. Validation checks confirmed that the import was successful, the filtering was correct, the expected date ranges were met, and the key fields were consistent across the filtered subset.

**3.3 DATA PREPROCESSING**

The preprocessing pipeline converts raw U.S. temperature information into a structured, gap-free time series that is ideal for forecasting. We prepare the data for feature engineering and model building by standardizing dates, selecting locations, filling in gaps, and removing outliers. The important steps include:

* **Date Parsing and Type Conversion**: We started by combining the separate Year, Month, and Day columns into a single Date field and casting it to pandas' datetime type. This process produced a continuous, gap-free time index, which is required for chronological analyses such as rolling averages and seasonal decomposition, while also allowing for efficient time-series operations. Records that failed to parse due to invalid or missing date components were eliminated, ensuring rigorous temporal continuity and data integrity throughout the dataset.
* **Filtering and Deduplication**: To narrow our focus, we began by keeping only rows where Country equals “United States,” thereby preserving data for every U.S. state and city originally present. We then removed any unintended duplicates by removing rows that had previously had the combination of Date, State, and City. We eliminated bias in subsequent models by maintaining only the first record of each unique triplet, ensuring that each location's daily temperature shows exactly once.
* **Missing Value Handling**: Once the data was aligned and deduplicated, we verified each column for null values. After eliminating inaccurate dates, we discovered no missing values in the essential temperature column. To maintain the continuity of our time series, any residual gaps in other characteristics were filled, ensuring that model inputs remained sequential while avoiding forward-looking bias.
* **Outlier Detection and Removal**: We used the interquartile range (IQR) approach for the AvgTemperature field, avoiding values outside Q1 - 1.5×IQR to Q3 + 1.5×IQR. This filtering excluded implausible extremes, which were most likely caused by sensor or entry errors, while keeping true seasonal and geographical variations.

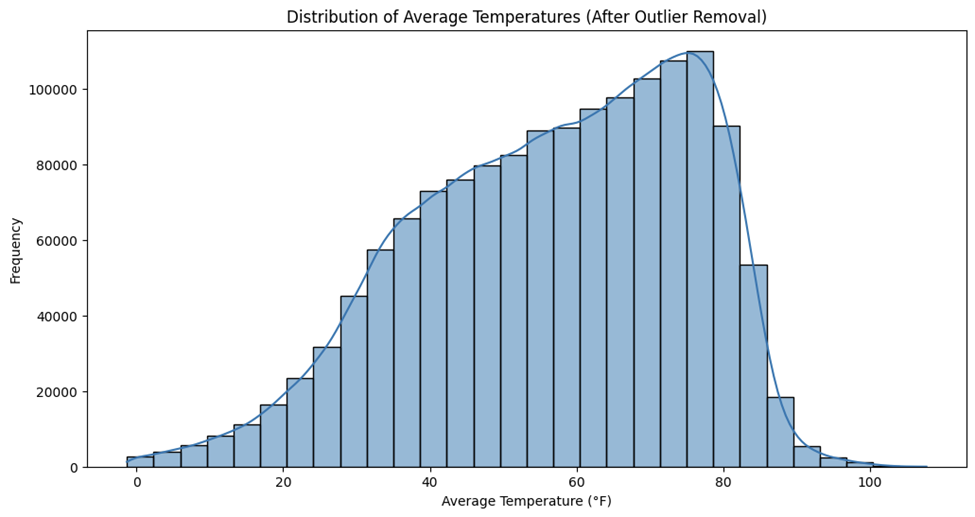
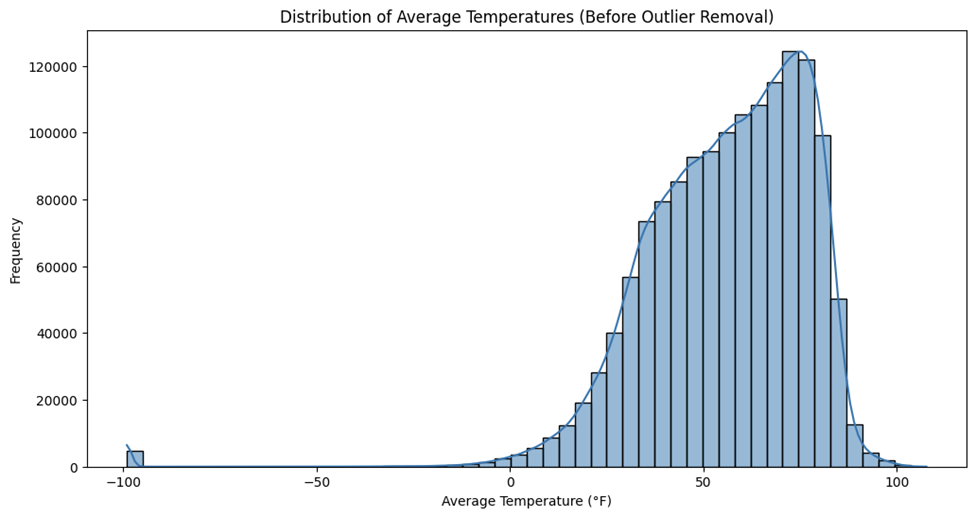


Figure 3.2a Figure 3.2b

Figure 3.2a shows that the original data had large tails and high spikes in temperatures that are well outside of anticipated seasonal ranges, which can bias model training and inflate error measures. After applying the IQR filter, Figure 3.2b shows a more concentrated, bell-shaped distribution with realistic minimums and maximums. By eliminating these implausible extremes, we limit variance and keep algorithms from "chasing" noise. The cleaned distribution produces more stable parameter estimates and speedier convergence, resulting in better forecast accuracy and interpretability.

**3.4 FEATURE ENGINEERING**

With the clean, U.S. temperature dataset in hand, we developed new predictors to help models capture short- and long-term patterns, seasonality, and location effects. The major steps were:

* **Lag Features**: We defined lagged temperature columns at 1-day, 7-day, and 30-day intervals (Temp\_Lag\_1, Temp\_Lag\_7, Temp\_Lag\_30). These qualities allow models to learn direct autocorrelation, which is how yesterday's, last week's, and last month's temperatures influence current values.
* **Rolling averages**: To smooth out day-to-day noise, we calculated 7-day and 30-day rolling means (Rolling\_7, Rolling\_30). These moving averages emphasize broader trends and seasonal shifts, allowing models to distinguish between brief spikes and substantial patterns.
* **Seasonal and Calendar Indicators**: We used the month to generate a Season label (Winter, Spring, Summer, Fall), and then one-hot encoded each season to create a binary feature. We also removed day-of-week, day-of-month, and week-of-year to account for the weekly cycles and monthly impacts that are common in temperature data.
* **Categorical Encodings**: The State and City columns were labeled with numeric codes to preserve regional distinctions while keeping feature dimensions reasonable for tree-based and deep learning models.
* **Scaling and Normalization**: Raw temperature values range widely from sub-freezing lows to summer highs, which can impede or destabilize learning, particularly in gradient-based models such as LSTMs. To overcome this, we used two common rescaling techniques:

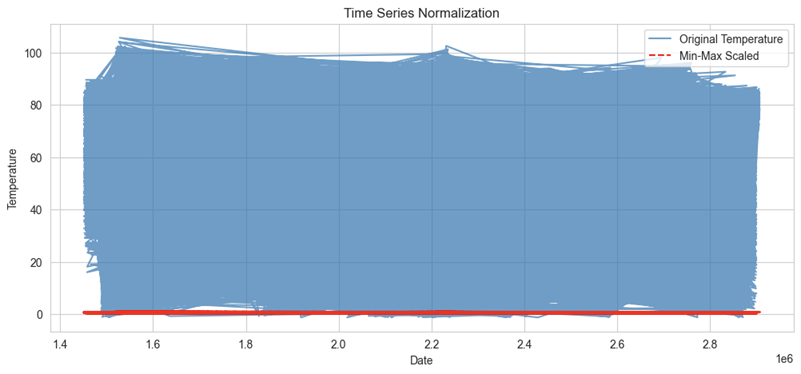
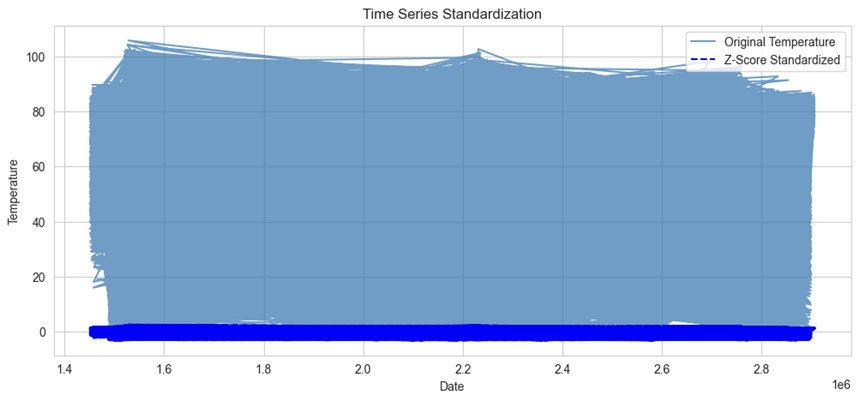
 

Figure 3.4a Figure 3.4b

Min-Max Normalization rescales all temperature values. As illustrated in Figure 3.4a, this transformation keeps the daily and seasonal oscillations structure while limiting all values to 0 and 1. Models like the LSTM benefit from avoiding gradient explosions induced by large inputs.

Z-score standardization centers temperatures around zero and scales them according to their standard deviation. Figure 3.4b shows that most daily values fall within ±2σ of the mean, reducing severe outliers and enhancing numerical stability. Standardized inputs often allow neural networks to converge more quickly and reliably.

* **Seasonal Decomposition**: To help models distinguish smooth, long-term shifts from repeating cycles and random noise, we decomposed the daily temperature series into three components: Trend, Seasonality, and Residual.

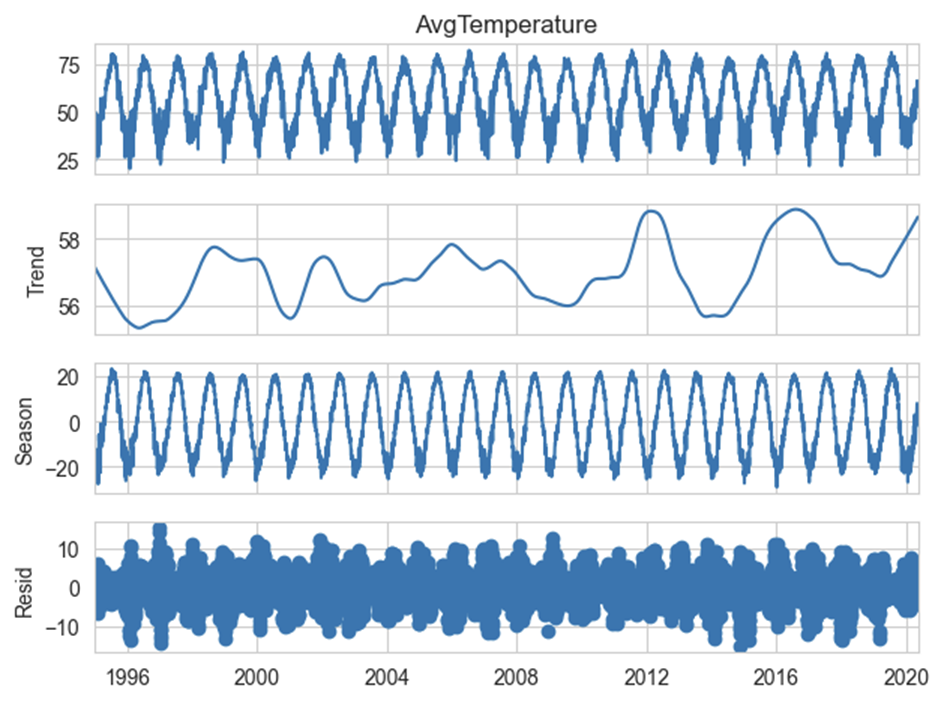
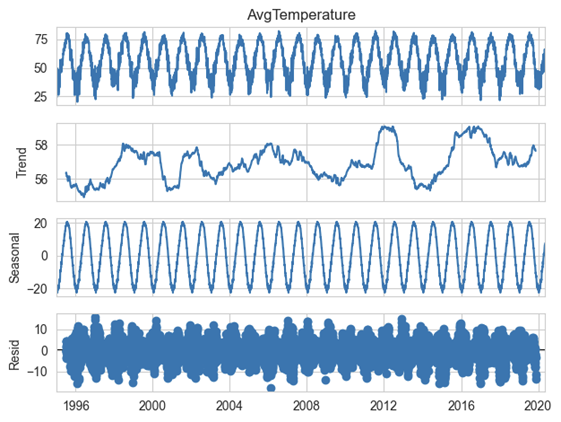


Figure 3.4c Figure 3.4d

In Figure 3.4c (Additive Decomposition), the trend panel traces the smooth, long-term trend in average temperatures, revealing gradual warming or cooling shifts over years, while the Seasonality panel isolates the regular annual cycle—summer highs and winter lows—and the Residual panel highlights day-to-day anomalies that are unexplained by trend and seasonality. Figure 3.4d (STL Decomposition) refines this further: its LOESS-smoothed trend flexibly follows gradual shifts and abrupt seasonal transitions, the seasonal component cleanly extracts the repeating yearly pattern, and the residuals panel displays only irregular deviations and noise after trend and seasonality are removed. By extracting these components as features, we provide models with direct access to the underlying climate signals, allowing them to focus on learning residual anomalies rather than recreating large seasonal or trend patterns from scratch.

**3.5 DATA MODELLING**

To capture the full complexity of temperature dynamics, our forecasting pipeline combines traditional time-series approaches with current machine learning. We evaluated five models: ARIMA and SARIMA for linear trends and seasonality, Random Forest and XGBoost for nonlinear interactions in manufactured features, and LSTM networks for learning long-term dependencies in sequential data for temperature forecasting. Each model is trained using our cleaned U.S. temperature series, with hyperparameters modified by automated methods (auto-ARIMA, grid search, and early stopping). Performance is compared using MAE, RMSE, and R². This diversified approach guarantees that we strike the right mix between interpretability, accuracy, and computing efficiency.

**3.5.1 ARIMA**

The Autoregressive Integrated Moving Average (ARIMA) model is a traditional statistical method that captures linear correlations in time series data by combining three components:

* Autoregression (AR) involves regressing a series on its own lagged values.
* Integration (I) is the process of differencing the series d times to attain stationary behavior.
* Moving Average (MA) represents the error term as a linear mixture of previous forecast errors.

ARIMA(p,d,q) refers to an ARIMA model in which p is the order of the autoregressive part, d is the degree of differencing, and q is the order of the moving-average part. Temperature data show substantial autocorrelation and slow trends, making ARIMA an appropriate benchmark. Its interpretability and well-understood behavior allow us to evaluate the added value of more advanced machine learning and deep learning technologies. ​

**APPLICATION:** We utilized the pmdarima library's auto\_arima procedure to automatically choose the ideal (p,d,q) parameters for each U.S. city's smoothed temperature series, searching p,q=0 to 4 and using a stepwise approach to balance accuracy and computing expense. The selected parameters were saved and then used to fit individual ARIMA models using statsmodels.tsa.arima. ARIMA. Each city's model then generated a 10-day forecast, which was stored alongside observed values for review and display.

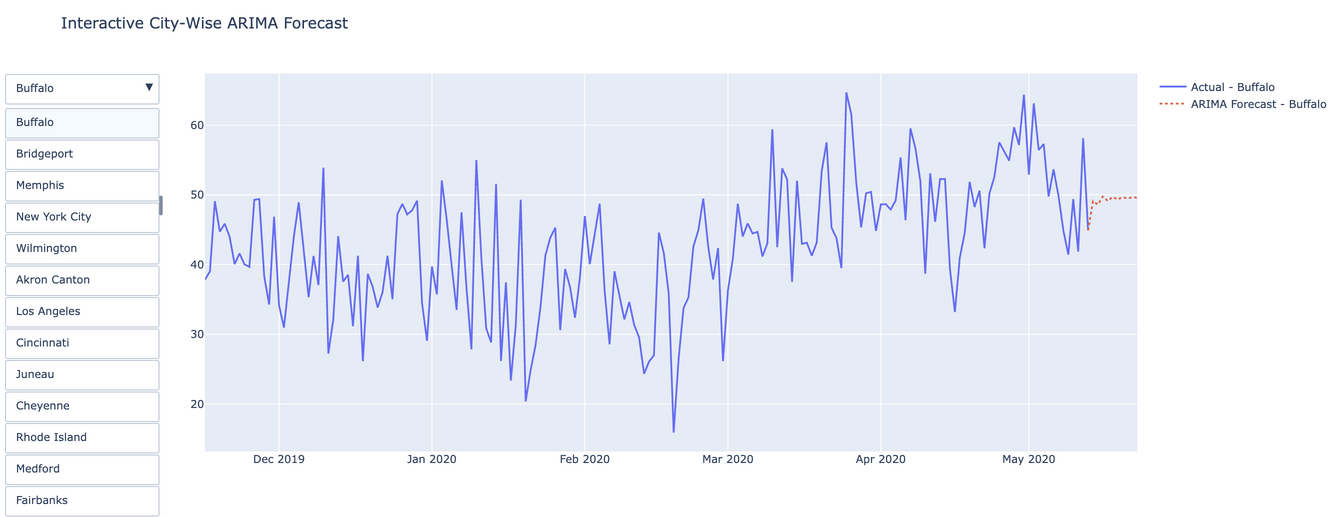
****

Figure 3.5.1a

This interactive Plotly graphic allows users to select any US city from a dropdown menu and compare the last 180 days of observed temperatures (solid line) with a 10-day ARIMA forecast (dotted line). By dynamically rotating cities, stakeholders may investigate location-specific performance, identifying areas where ARIMA shines or struggles with local anomalies, making it an effective tool for targeted operational planning.

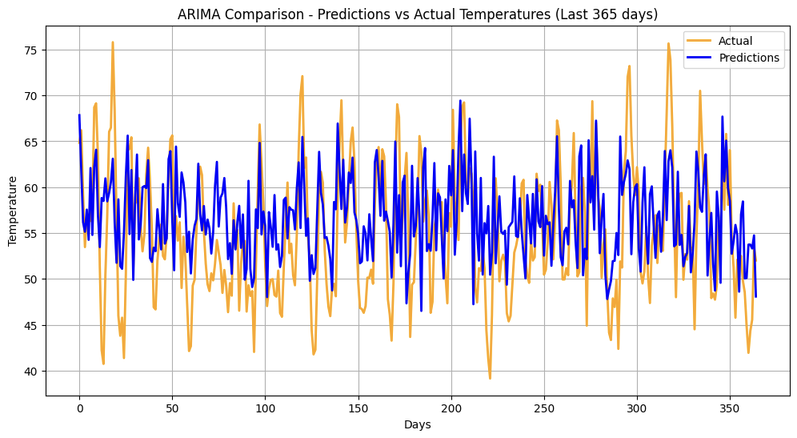


Figure 3.5.1b

Figure 3.5.1b ARIMA Forecast versus Actual Temperatures. This line chart compares the ARIMA model's one-step-ahead predictions (blue line) to the actual smoothed temperature series (orange line) for the last 365 days. It demonstrates how the ARIMA(p,d,q) structure captures the broad seasonal ebb and flow, tracking summer peaks and winter troughs, while occasionally falling behind abrupt anomalies. The tight alignment for the majority of the year highlights ARIMA's ability to simulate linear dependencies and regular cycles.

**3.5.2 RANDOM FOREST**

Random Forest is an ensemble learning method that creates numerous decision trees from bootstrapped data and averages their outputs to increase forecast accuracy and prevent overfitting. Each tree sees a random selection of features, allowing it to capture complicated, nonlinear relationships in the data.

Random Forest excels at handling high-dimensional, mixed-type inputs, such as rolling averages, lagged temperatures, and label-encoded geographic codes, without requiring rigorous distributional assumptions. By averaging several decorrelated trees, it decreases overfitting to noisy fluctuations while also capturing nonlinear interactions between attributes. The built-in feature importance measure identifies the most important predictors for accuracy, driving further feature engineering refinements. Random Forest is both accurate and efficient for big, multi-city datasets because of its ability to train several trees in parallel.

**APPLICATION:** We trained a Random Forest regressor on our cleaned U.S. temperature dataset, employing features such as one-day, one-week, and one-month lagged temps, 7-day and 30-day rolling averages, day-of-week, day-of-year, and week-of-year, as well as label-encoded state and city. To avoid look-ahead bias, the data were split 80/20 chronologically. Using 100 trees (n\_estimators=100) and parallel processing, the model learnt both short-term variations and longer seasonal trends. After training, we created 10-day forecasts for each city and evaluated them using RMSE, MAE, and R².

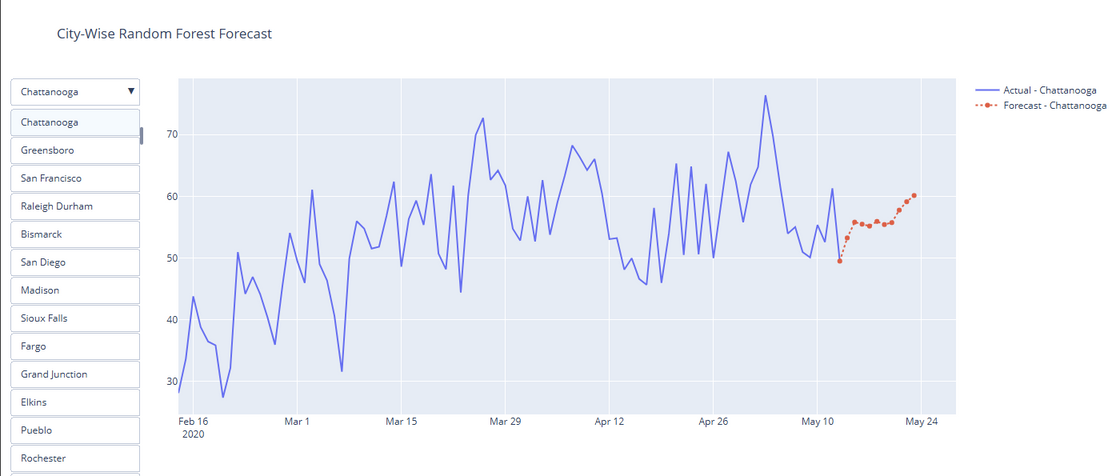


Figure 3.5.2a

Interactive City-Wide Random Forest Forecast. This interactive map shows the previous 90 days of genuine temperatures (solid blue) for a specific city, as well as a 10-day Random Forest forecast (red dotted). Users can select any city from the dropdown to see how well the model retains recent dynamics and predicts short-term temperature changes, allowing stakeholders to evaluate location-specific forecast reliability in a single dynamic view.

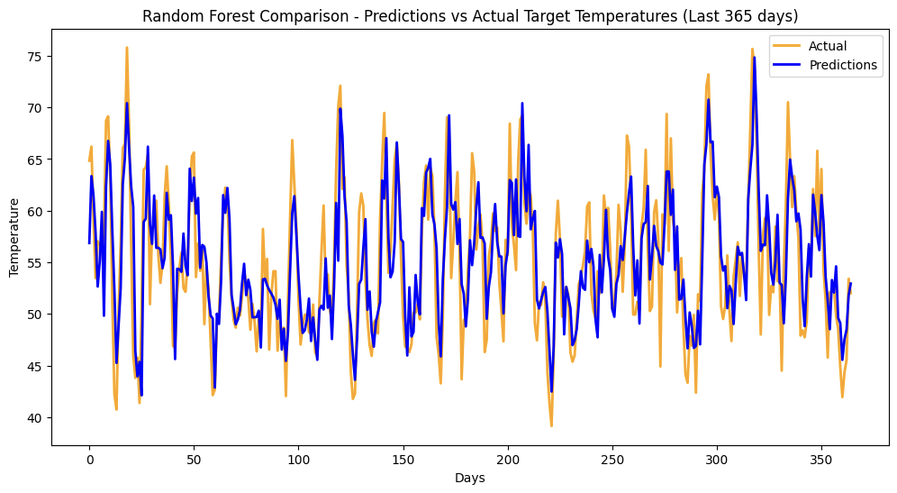


Figure 3.5.2b

Random Forest Predictions vs. Actual Temperatures (Last 365 Days). This line chart compares the Random Forest's smoothed temperature forecasts (blue) to the actual observed temperatures (orange) for the most recent year. The two lines closely track each other through seasonal peaks and troughs, illustrating the model's ability to learn both slow trends and short-term fluctuations, while occasional deviations indicate periods when localized anomalies temporarily outperform the ensemble's averaging impact.

**3.5.3 XGBOOST**

XGBoost (eXtreme Gradient Boosting) is an improved gradient-boosting system that successively constructs ensembles of decision trees, with each tree repairing the mistakes of its predecessor. It uses second-order gradient information to speed up convergence, regularization to prevent overfitting, and a histogram-based technique for rapid, parallelized tree learning. These design decisions make XGBoost both efficient and accurate on huge datasets.

Temperature forecasting relies on complicated, nonlinear relationships between delayed temperatures, seasonal cycles, and regional considerations. XGBoost excels in this area by automatically handling missing information, weighting feature relevance, including built-in regularization to avoid overfitting to noise, and allowing variable hyperparameter adjustment for a wide range of climatic situations. Its histogram-based tree builder and parallelized training combine to give exceptionally accurate forecasts even on large, multi-city datasets.

**APPLICATION:** We trained an XGBRegressor on our cleaned U.S. temperature dataset with the same feature set as Random Forest, which included 1-day, 7-day, and 30-day delayed temps, 7-day and 30-day rolling averages, day-of-week, day-of-year, and week-of-year, as well as label-encoded state and city codes. The model was set up with 500 trees (n\_estimators=500), a 0.05 learning rate, using the histogram-based tree technique for speed. To prevent look-ahead bias, we fit on an 80/20 chronological split, saved the trained model, and generated both in-sample and 10-day forward projections for each city, allowing us to directly compare forecast and actual temperatures.

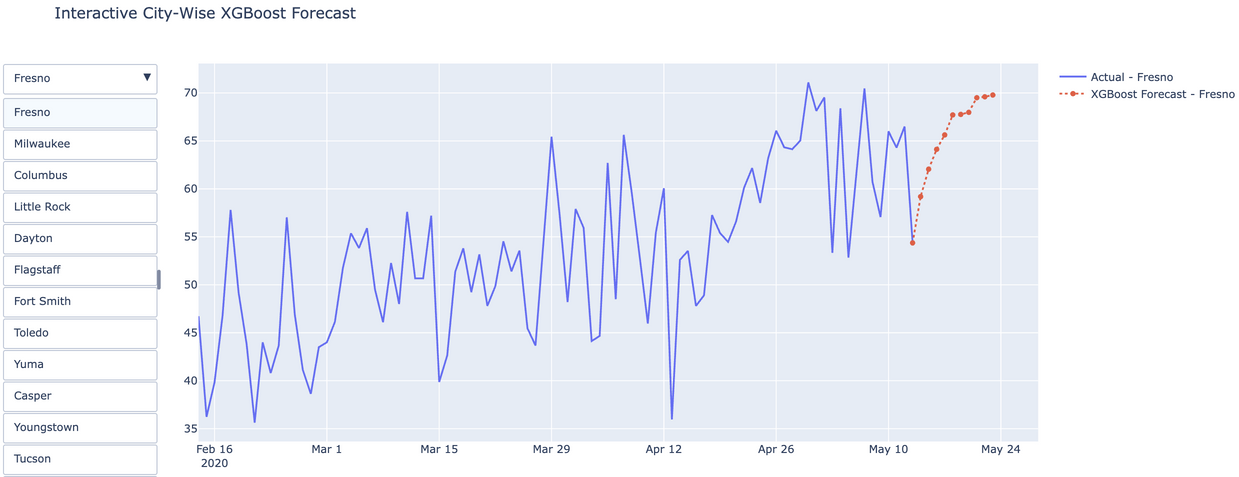


Figure 3.5.3a

Interactive city-wise XGBoost forecast. This Plotly chart allows users to select any city from a menu and compare the most recent 90 days of real temperatures (solid blue) to a 10-day XGBoost forecast (red dotted). By dynamically switching between cities, stakeholders can investigate location-specific forecast accuracy and acquire confidence in the model's short-term estimates across varied climates.

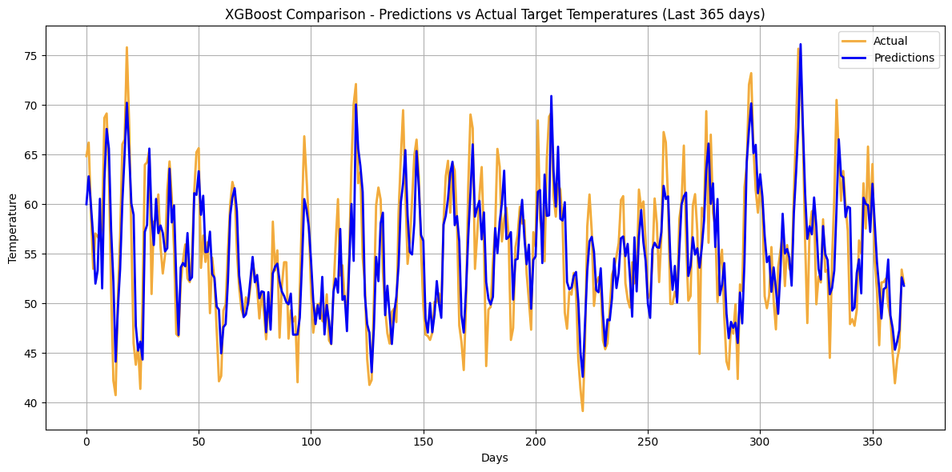


Figure 3.5.3b

XGBoost Predictions vs. Actual Temperatures (last 365 Days). This line plot overlays the XGBoost model's 365-day forecasts (blue dashed) with the observed smoothed temperature data (orange). The close alignment of seasonal peaks and troughs indicates XGBoost's capacity to learn both recurrent cycles and short-term changes, while slight divergences indicate when rapid temperature swings outperform the ensemble's incremental updates.

**3.5.4 LSTM**

Long Short-Term Memory (LSTM) is a recurrent neural network that can learn both short- and long-term relationships in sequential input. Unlike traditional RNNs, LSTMs use gated memory cells, which include input, forget, and output gates, to manage the flow of information, allowing the network to selectively recall or discard past values over many steps. This architecture solves the vanishing-gradient problem, allowing LSTMs to simulate complicated temporal patterns, such as seasonal cycles and unexpected anomalies, in time series like daily temperature records. Their ability to keep and update an internal "state" allows for more accurate, context-aware forecasting across extended horizons.

Unlike tree-based or linear models, LSTMs can explicitly express complicated, nonlinear dependencies over multiple time steps. In temperature forecasting, this entails understanding how patterns from a month ago or even earlier affect today's temperature and adjusting to rapid changes. Their innate memory power allows them to capture seasonal cycles and unexpected events without substantial manual feature engineering.

**APPLICATION:** We then used Z-score standardization to scale all LSTM inputs, which included smoothed temperature, lagged data, rolling averages, seasonal flags, calendar features, and normalized temperature. NumPy stride tricks were used to generate 30-day sequences, which yielded 3D tensors for model input. Our Keras Sequential design combines two LSTM layers (100 units returning sequences, then 50 units), each with a 20% dropout, and dense layers to provide one-step forecasts. We trained for up to 15 epochs with early pausing and learning rate decrease on validation loss, and then inverted the scaling to retrieve the original temperature units for evaluation.

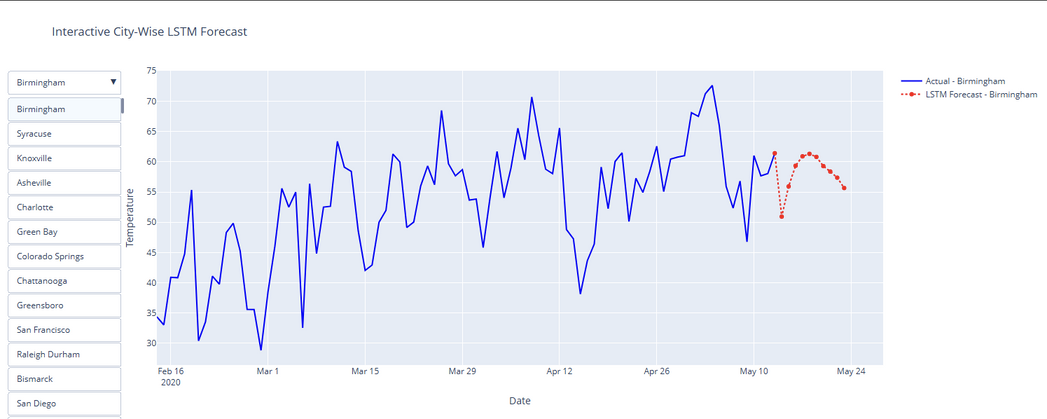


Figure 3.5.4a

Interactive City-Wise LSTM Forecast. This Plotly visualization allows users to select a city from a menu and examine its previous 90 days of recorded temperatures (solid blue) alongside a 10-day LSTM forecast. By dynamically rotating cities, stakeholders can investigate how the model tailors its forecasts to each location's distinct climate patterns, thereby measuring short-term reliability across varied geographies.

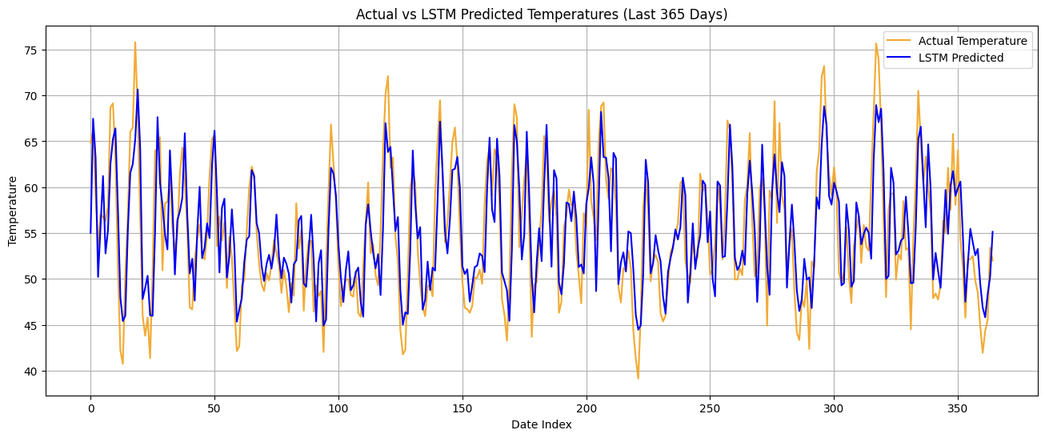


Figure 3.5.4b

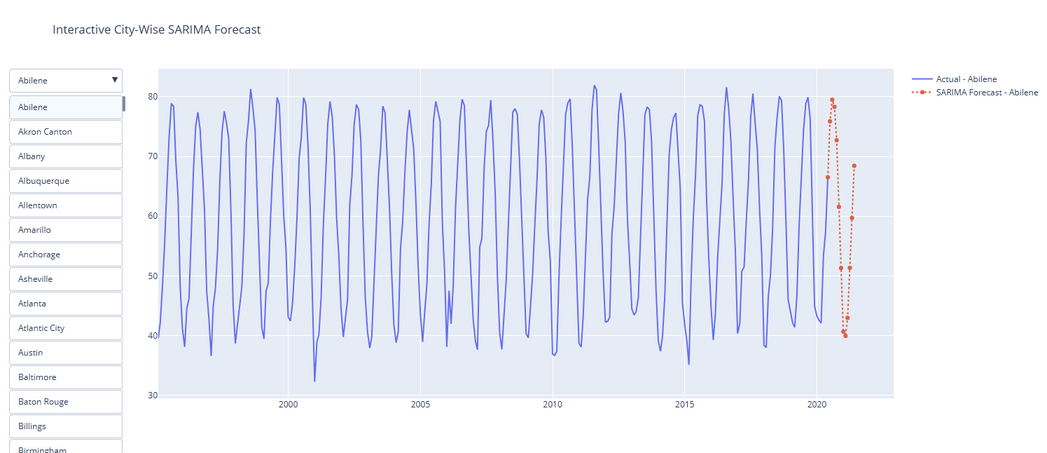
Actual vs. LSTM‐Predicted Temperatures (Last 365 Days). This line chart compares the LSTM model's one-step-ahead predictions (blue line) with the actual smoothed temperature series (orange line) for the most recent year. The close alignment between summer peaks and winter troughs indicates the network's capacity to detect both recurrent seasonal cycles and shorter abnormalities. Small lags during abrupt temperature fluctuations indicate when the model alters its internal state to accommodate quick changes.

**3.5.5 SARIMA**

Seasonal ARIMA (SARIMA) improves on the conventional ARIMA model by including seasonal variables such as a seasonal autoregressive component (P), seasonal differencing (D), and a seasonal moving-average component (Q), in addition to the non-seasonal (p,d,q) parameters. SARIMA(p,d,q)×(P, D, Q)m, where m is the seasonal period, captures both short-term correlation and long-term cycles, such as the annual temperature rhythm.

Non-seasonal ARIMA models cannot properly reflect the temperature's strong annual cycle, which includes summer highs and winter lows. SARIMA explicitly predicts these seasonal influences, which improves forecast accuracy over longer periods. Fitting non-seasonal and seasonal components balances slow trends, noise, and periodic patterns in climate data.

**APPLICATION:** We aggregated daily U.S. temperatures to monthly averages for each city, then used auto\_arima to pick optimal ordering for both non-seasonal and seasonal terms (P, D, Q), with m=12. Each city's SARIMAX model was fitted to its monthly series, generating 12-month projections. We examined model performance across all cities, using SARIMA as a baseline for our machine learning and LSTM techniques.

Figure 3.5.5a

A dropdown menu in the interactive display allows viewers to select any city in the United States and compare its historical monthly temperature averages (solid blue) to the SARIMA model's 12-month forecast (red dotted). As you move between cities from coastal hubs to interior plains, you can observe how vividly defined the seasonal cycles are in certain areas and more muted in others. This dynamic study also identifies areas where the model's confidence is waning, forecast uncertainty frequently peaks in places with unpredictable weather events, informing stakeholders where to exercise caution or seek further data.

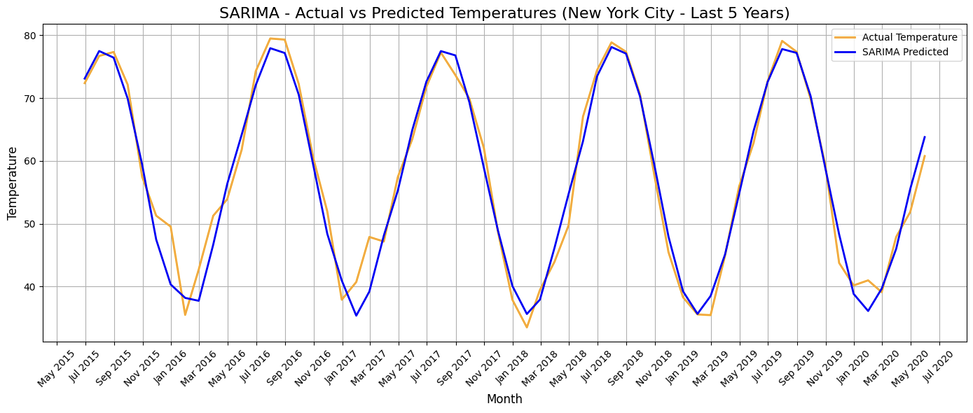


Figure 3.5.5b

This plot compares the SARIMA model's fitted monthly forecasts (blue) to the actual, three-month-smoothed temperature averages (orange) for New York City from May 2015 to May 2020. Notice how the peaks in July and troughs in January are nearly identical, illustrating the model's capacity to learn both the amplitude and timing of annual cycles. Subtle phase shifts, such as a slightly earlier summer peak in 2018 or a later winter dip in 2017, show times when real-world anomalies or emergent patterns temporarily exceed the simple seasonal framework.

**3.6 VISUALIZATIONS**

To bring our forecasts and metrics to life, we created two interactive dashboards that allow users to compare model performance over time, region, and error dimensions. The first dashboard compares anticipated and actual temperature traces for individual cities, while the second analyzes error statistics (MAE, RMSE, R²) across models, months, and US locations. Together, these visual tools help planners and analysts gain intuitive, data-driven insights.

**3.6.1 FORECAST COMPARISON DASHBOARD**

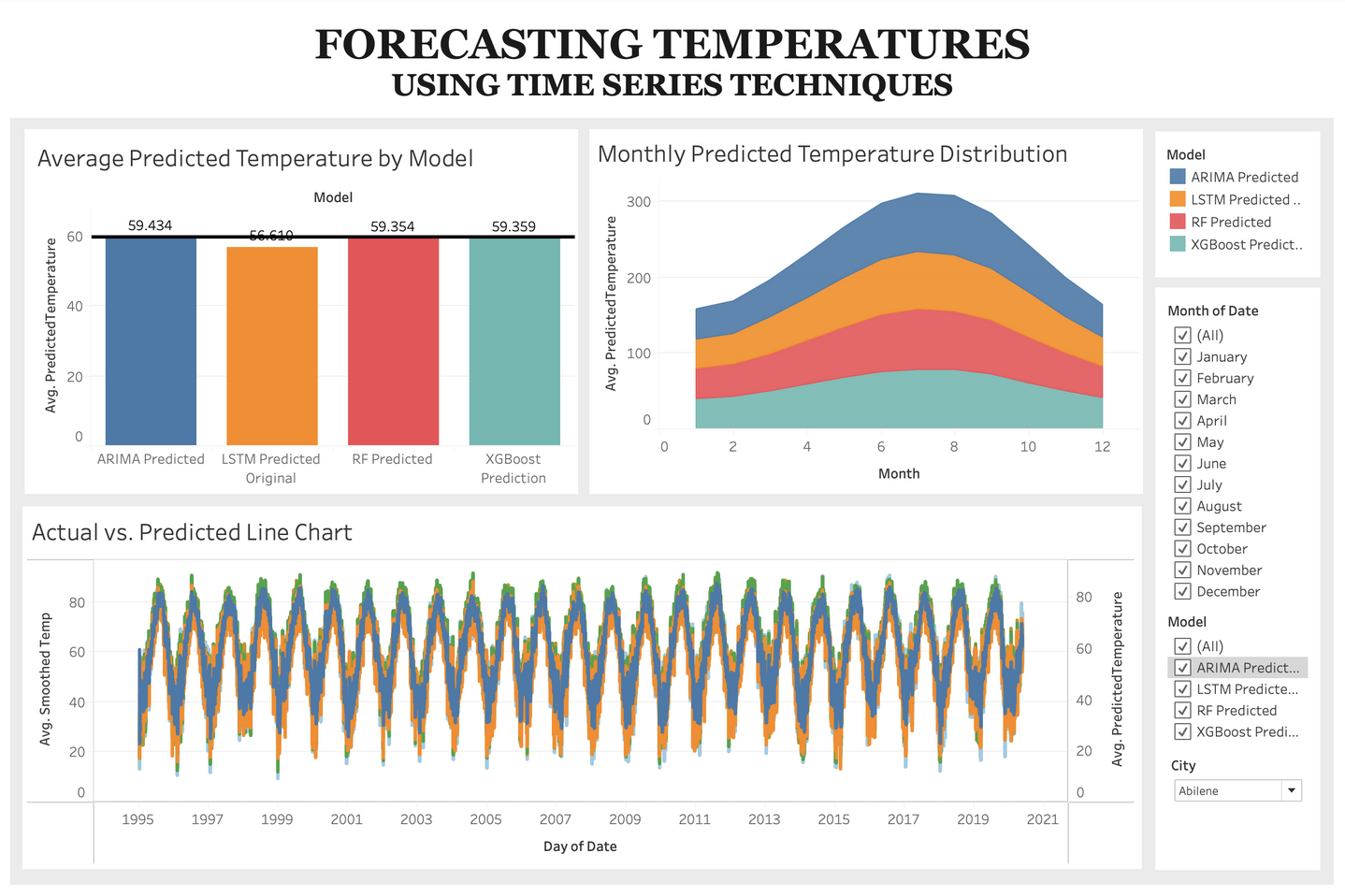
****

Figure 3.6.1a

**Average Predicted Temperature by Model**: A bar chart comparing each model's mean forecast temperature throughout the whole period, indicating systematic overestimation or underestimation and relative baseline shifts, allowing users rapid insight into model bias and performance.

**Monthly Forecasted Temperature Distribution**: A stacked area chart of each model's average forecast temperature per month, emphasizing seasonal amplitude variances and peak timing. Hover tooltips display exact monthly numbers, allowing for comparison of seasonal biases.

**Actual vs. Predicted Line Chart**: A multi-line figure compares actual smoothed temperatures to forecasts from ARIMA, Random Forest, XGBoost, LSTM, and SARIMA, demonstrating each model's tracking accuracy, divergence moments, and seasonal consistency across time.

Overall, the Forecast Comparison Dashboard allows for the evaluation of model biases and seasonal performance across series, months, and cities. Bar, area, and line charts show each model's strengths and drawbacks. Interactive filters for model selection, month range, and city enable targeted, actionable analysis based on temporal or geographic demands.

**3.6.2 ERROR ANALYSIS DASHBOARD**

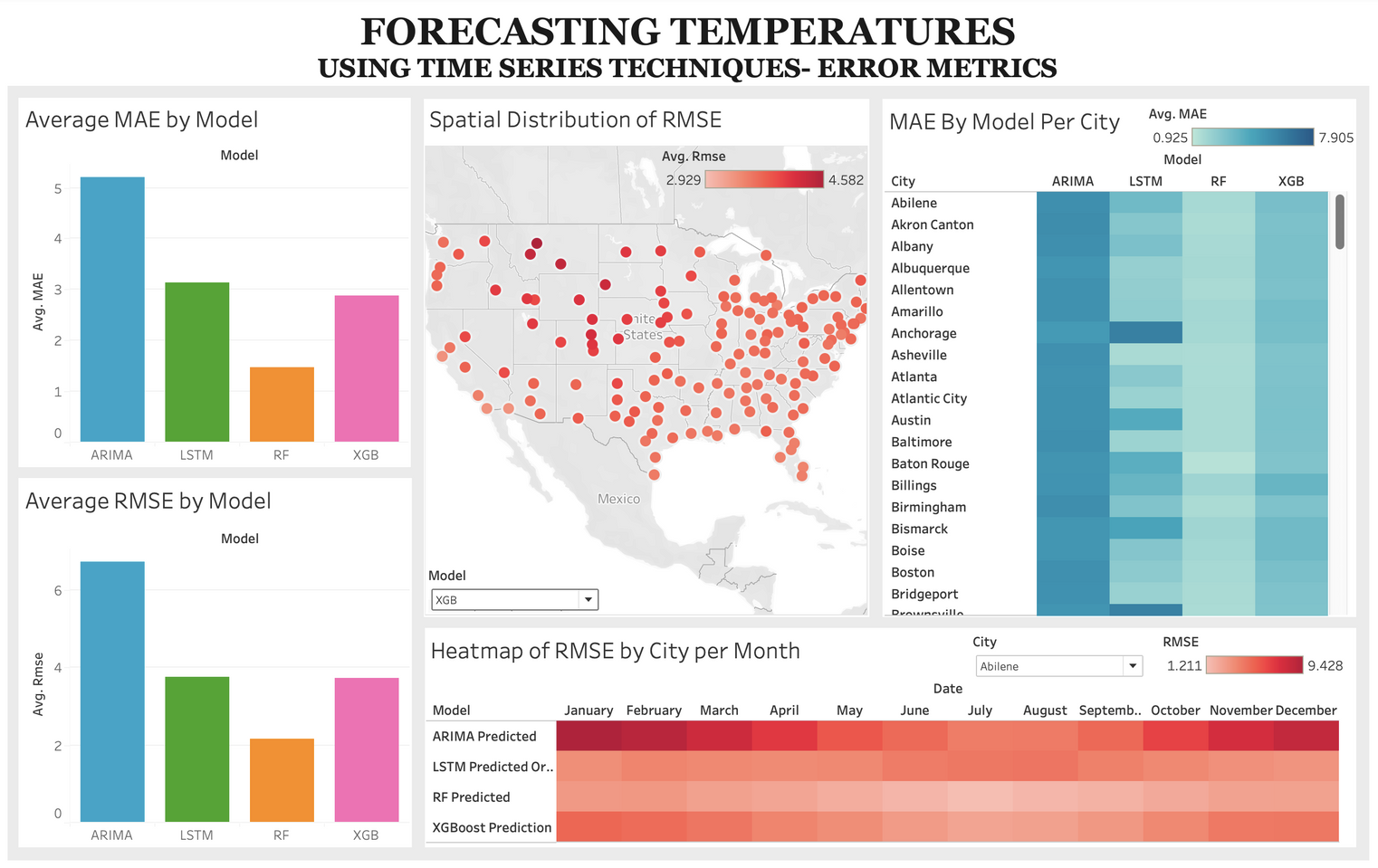
****

Figure 3.6.2a

**Average MAE by Model**: A bar chart compares the average mean absolute error (MAE) of each model across all cities in the United States. Lower bars represent more accurate point projections, emphasizing Random Forest's superior overall performance.

**Average RMSE by model**: A bar chart displaying each model's root-mean-square error (RMSE) averaged across all sites. It demonstrates that Random Forest and LSTM achieve lower overall variance in forecast errors than ARIMA and XGBoost.

​​**Spatial distribution of RMSE**: A geographic map depicts each city as a circle, with the size and color intensity reflecting its RMSE for the chosen model. Coastal and mountainous locations have darker, larger markers, suggesting greater forecast uncertainty, whereas inland plains have smaller, lighter circles, revealing more dependable predictions in less variable climates. This spatial view indicates where localized model changes or further data collection may be most necessary.

**MAE by Model per City**: This vertical heatmap lists each city in the United States along the y-axis and includes a column for each model's mean absolute error (MAE). Cell color intensity, ranging from light teal (low MAE) to deep blue (high MAE), instantly identifies which models suffer in specific environments. ARIMA, for example, produces darker cells in northern towns with unpredictable winters, whereas Random Forest consistently produces milder shading throughout the majority of sites.

**Heatmap of RMSE by City Per Month**: This matrix heatmap shows months (January-December) on the x-axis and model names on the y-axis, with each cell's color ranging from light coral (low RMSE) to deep red (high RMSE), signifying the model's inaccuracy for the given month-city combination (where a city filter is used).

Overall, the Error Metrics Dashboard converts raw forecast evaluations into a usable, interactive interface. Users may quickly compare MAE and RMSE distributions, identify regional regions of high error using the U.S. map, and dig down into city- and month-specific heatmaps to reveal seasonal vulnerabilities. Filters for model type and city allow for focused analysis, leading to targeted changes such as refining feature engineering or retraining specific models in underperforming locations.

1. **RESULTS AND ANALYSIS**

Our review evaluates five forecasting approaches: ARIMA, SARIMA (monthly), Random Forest, XGBoost, and LSTM, using three essential metrics:

* Root Mean Square Error (RMSE),
* Mean Absolute Error (MAE), and
* Coefficient of Determination (R²).

We evaluate each model's accuracy using city-level fitted data and performance.

### **4.1 MODEL PERFORMANCE COMPARISON**

| **MODEL** | **RMSE** | **MAE** | **R^2** |
| --- | --- | --- | --- |
| XGBOOST | 3.7438 | 2.8738 | 0.9470 |
| LSTM | 3.9826 | 3.1702 | 0.9400 |
| RANDOM FOREST | 3.9175 | 2.9997 | 0.9394 |
| ARIMA | 6.7225 | 5.1993 | 0.8211 |
| SARIMA (Monthly) | 9.8122 | 3.8574 | 0.4978 |

Table 4.1.1 Average error metrics across all U.S. cities ​

XGBoost: With RMSE 3.7438, MAE 2.8738, and R² 0.9470, XGBoost achieves improved accuracy and variance explanation by capturing complicated nonlinear interactions and seasonal trends. It also benefits from regularization and efficient tree pruning across all cities.

LSTM: The LSTM network model, with RMSE 3.9826, MAE 3.1702, and R² 0.9400, excels at capturing long-term temporal dependencies. Its gated memory cells adjust to seasonal trends, but they must be carefully tuned for optimal convergence.

Random Forest: Random Forest, with RMSE 3.9175, MAE 2.9997, and R² 0.9394, balances predictive strength and stability through an ensemble of decorrelated trees that handle noise robustly and effectively, with minimal adjustment.

ARIMA: The ARIMA model has an RMSE of 6.7225 and an MAE of 5.1993, indicating that it accurately captures linear trends and seasonality. However, it suffers from nonlinear patterns and abrupt anomalies, demonstrating its limited adaptability when compared to ensemble approaches.

SARIMA: Monthly SARIMA records: RMSE 9.8122, MAE 3.8574, R² 0.4978. It accurately mimics annual cycles but loses daily detail due to aggregation, resulting in lower accuracy and more information loss than daily techniques.

**4.2 CITY-LEVEL FORECAST ERROR VARIATION**

Forecast errors change systematically with local climate for all four modelling systems. Random Forest yields the lowest RMSE values in San Juan (≈1.21 °F) and Daytona Beach (≈1.33 °F)—warm, coastal locales with modest variability—while Rapid City, Helena, and Great Falls (semi-arid, mountainous regions) show the highest errors (≈2.57-2.61 °F). XGBoost performs well in San Juan, Honolulu, and San Diego (RMSE ≈2.93-3.32 °F) but poorly in Helena, Billings, and Great Falls (RMSE ≈4.40-4.58 °F). LSTM has minimal inaccuracy in Asheville and Roanoke (≈1.87-2.09 °F), but difficulties in high-latitude or insular climates like Honolulu, Fairbanks, and San Juan (≈7.84-8.67 °F). ARIMA's lowest RMSE occurs in Honolulu and San Juan (≈6.00-6.07 °F), with the biggest error in the industrial Midwest, Dayton, Youngstown, and Cleveland (≈7.15-7.19 °F).

These patterns show that models perform better in stable, coastal climates, whereas mountainous and continental locations present higher forecasting difficulties.

1. **CONCLUSION**

**5.1 CONCLUSION**

Accurate temperature forecasting is crucial for agriculture, energy management, and urban planning. In this capstone, we created an end-to-end pipeline from raw data ingestion to preprocessing, feature engineering, model training, and interactive visualization to convert daily U.S. temperature records into actionable insights. Rather than depending on a single strategy, we blended classical time-series methods (ARIMA and SARIMA), ensemble learners (Random Forest and XGBoost), and deep learning (LSTM) to balance interpretability, accuracy, and computing economy.

Our findings suggest that modern ensemble approaches, particularly gradient-boosted trees, achieve the highest daily forecast accuracy with low tuning effort, whereas neural networks such as LSTM excel at capturing longer-term dependencies when enough compute resources are available. Traditional techniques, while less precise, are nevertheless helpful as transparent benchmarks and in cases where model simplicity and explainability are critical. We gave each model a comprehensive, context-aware feature set by engineering lagged data, rolling averages, seasonal indicators, and geographic encodings, resulting in performance gains across varied climates.

Beyond numerical metrics, we created two interactive dashboards that allow stakeholders to compare model predictions to actual temperatures, investigate where forecast mistakes are concentrated regionally and seasonally, and customize filters for certain locations, time ranges, and error indicators. This visual layer transforms statistical outputs into a decision-support tool, allowing users to drill down into forecasts, validate model choices, and better allocate resources, whether protecting crops from frost, balancing electricity supplies during heatwaves, or planning city heat mitigation strategies.

Overall, this project provides a completely repeatable framework, complete with open-source code and modular components, which can be tailored to various regions, expanded to include other meteorological variables, or linked into real-time forecasting systems. We lay the framework for data-driven temperature forecasting that fulfills the operational needs of a wide range of sectors and communities by combining rigorous models and intuitive visuals.

**5.2 PROJECT LIMITATIONS**

While our forecasting pipeline produces excellent results, several practical constraints and design decisions limit its current breadth and generalizability. First, computing resources limited our modeling granularity: fitting SARIMA on daily, city-level series was unacceptably slow and memory-intensive, leading us to aggregate data to monthly averages for seasonal modeling while sacrificing daily detail. To prevent unnecessary runtimes on large datasets, ensemble and deep-learning models needed to be carefully tuned in terms of batch sizes, tree counts, and early stopping conditions.

Second, our feature set is solely focused on temperature, ignoring other meteorological drivers such as humidity, wind speed, and air pressure that influence genuine weather dynamics. Without these exogenous inputs, models may underperform during unexpected events (for example, cold fronts or heat waves) in which non-temperature elements play a significant role. Third, data preparation decisions, such as IQR-based outlier removal and missing value backfilling, may accidentally eliminate legitimate extreme observations or induce small biases in sequence continuity. Although these processes stabilized training, they may obscure uncommon but significant irregularities.

Finally, our analysis is limited to cities in the United States between 1995 and 2019, which limits its geographic and temporal transferability. As climate trends change, models built on historical data may need to be retrained or augmented with more recent observations. Addressing these restrictions will be crucial for deploying strong, real-time forecasting systems in a variety of situations.

**5.3 FUTURE RESEARCH**

In the future, we plan to add external meteorological data (humidity, wind speed, and pressure), test multivariate time-series models (ARIMAX, vector autoregression), investigate hierarchical spatiotemporal frameworks, and incorporate Transformer-based designs for long-range relationships. We will use online learning for real-time forecasting and expand the pipeline to include global and extreme weather information. To properly quantify uncertainty, we will also consider ensemble stacking and probabilistic forecasting (quantile regression and Bayesian approaches). Finally, we intend to use automated updating mechanisms to retrain models using incoming data streams, ensuring adaptability to changing climate trends. We also plan to incorporate explainable AI approaches to increase model transparency and consumer trust.

1. **REFERENCES**

1. Bandara, K., Hyndman, R. J., & Bergmeir, C. (2021). MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. arXiv preprint arXiv:2107.13462.<https://arxiv.org/abs/2107.13462>

2. Kumar, A., Puri, S., Selokar, P., Bokre, R., & Talhar, Y. (2020). Forecasting of temperature by using time series analysis. International Journal of Research and Analytical Reviews (IJRAR), 7(1), 547–551. <http://www.ijrar.org/IJRAR2001081.pdf>

3. Luo, J., Zhang, Z., Fu, Y., & Rao, F. (2021). Time series prediction of COVID-19 transmission in America using LSTM and XGBoost algorithms. Results in Physics, 27, Article 104462. <https://doi.org/10.1016/j.rinp.2021.104462>

4. Meenal, R., Michael, P. A., Pamela, D., & Rajasekaran, E. (2021). Weather prediction using random forest machine learning model. Indonesian Journal of Electrical Engineering and Computer Science, 22(2), 1208–1215. <https://doi.org/10.11591/ijeecs.v22.i2.pp1208-1215>

5. Singh, R., Sherigar, S., & Singh, S. (2024). Temperature forecasting and analysis using linear and time-series models. Iconic Research and Engineering Journals, 7(8), 78–84.<https://www.irejournals.com/formatedpaper/1705460.pdf>

6. Ashraf, S., Ali, N., Khan, M. I., Niaz, A., & Masood, M. (2024). Weather forecasting: Using time series analysis under different stations of the UK. Migration Letters, 21(S9), 782–794.<https://www.sciencedirect.com/science/article/pii/S2211379721005775>

1. **APPENDIX**

**Appendix A1: Lag and Rolling Average Features.**

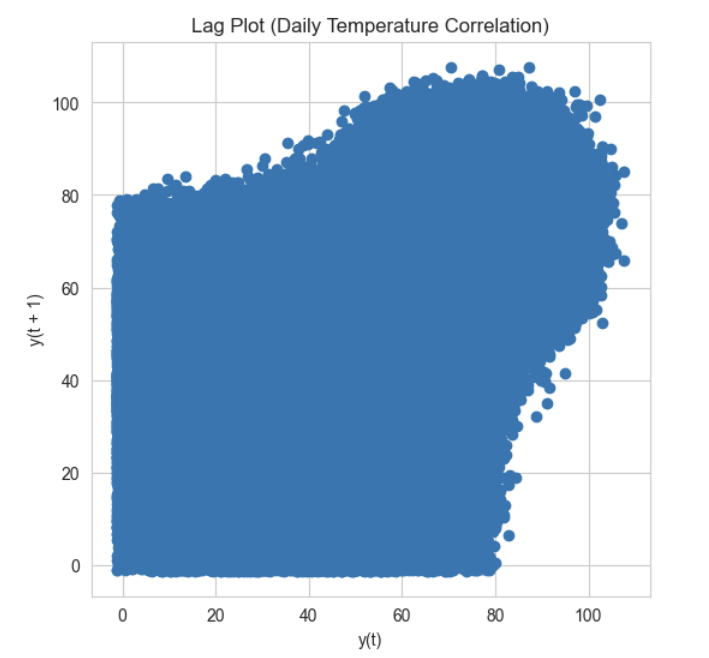
****

Figure A2.1. Lag Plot (Daily Temperature Correlation)

This scatter plot of y(t) vs. y(t+1) demonstrates a substantial positive autocorrelation, with most spots close to the diagonal. The tight cluster indicates that yesterday's temperature is a strong predictor for today, supporting the inclusion of one-day lag characteristics in our models.

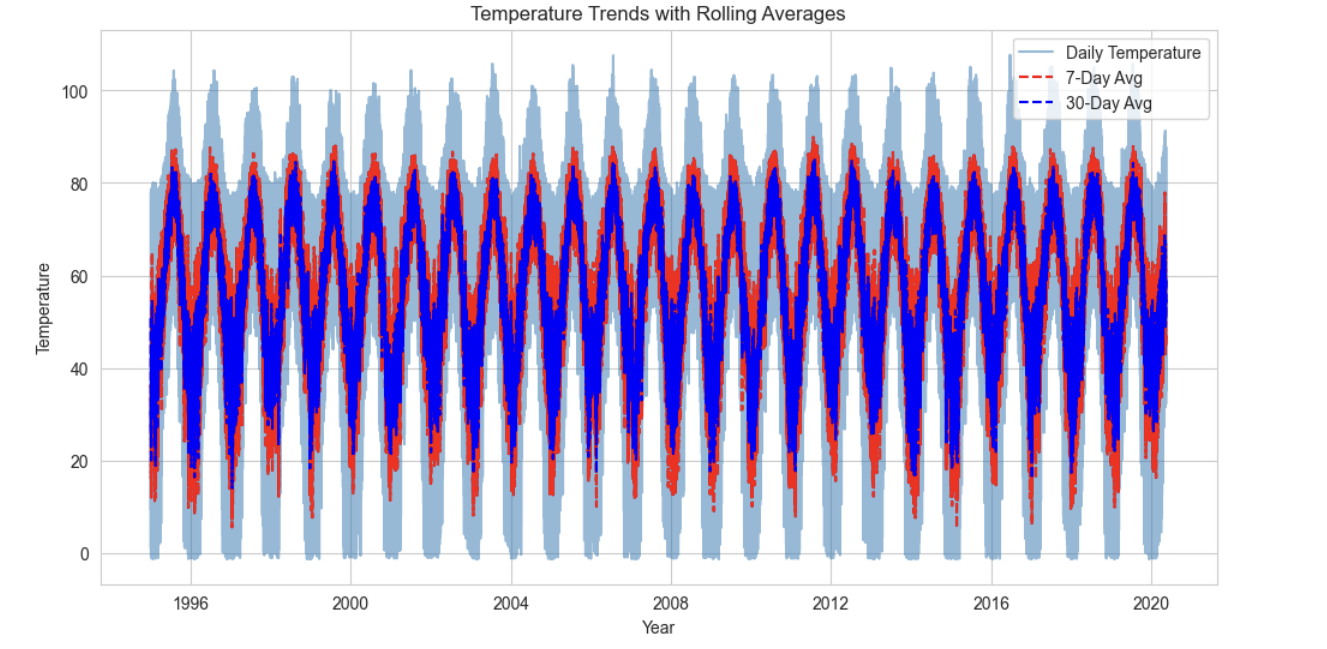
****

Figure A2.2. Temperature Trends with Rolling Averages

This time-series plot shows raw daily temperatures (light blue) overlaid with rolling averages for 7 days (red dashed) and 30 days (blue dashed). Rolling averages efficiently smooth out day-to-day noise, revealing underlying seasonal cycles and long-term trends that models can use to improve prediction stability.

**Appendix A2: Feature Correlation and Temporal Patterns.**

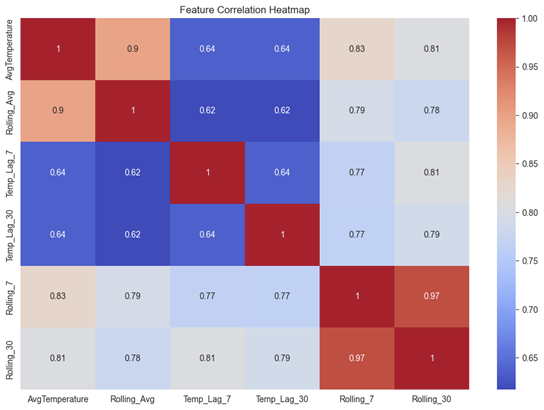


Figure A2: Feature Correlation Heatmap.

This matrix displays the Pearson correlation coefficients between our engineered features and the goal average temperature (°F):

Temp\_Lag\_7 and Temp\_Lag\_30 have a moderate (≈0.64) correlation with AvgTemperature and a significant (0.64) correlation with one another.

Smoothed Trends: Rolling\_7 and Rolling\_30 are virtually identical (0.97) and closely track AvgTemperature (≈0.83 and 0.81), making them suitable for trend smoothing.

Aggregate Smoothing: Rolling\_Avg (7-day smoothed series) has a very high correlation with AvgTemperature (0.90), demonstrating its relevance as a key predictor.

Figure A2 confirms that lagged and rolling-average features provide complementary signals, capturing short-term autocorrelation and longer seasonal trends. It also guides us to avoid redundant inputs, such as dropping Rolling\_7 or Rolling\_30 in leaner models.