Real-Time Weather Anomaly Detection System

Ahmet Doğukan Gündemir
October 2024

Table of Contents

Introduction

Weather Anomaly Detection

Machine Learning Integration

Using machine learning algorithms to analyze weather data and identify anomalies quickly.

Background and Motivation

Climate change has escalated the demand for precise weather monitoring systems to predict anomalies.

Objective of the Project

To develop an IoT-based system designed for real-time detection of weather anomalies.

Need for Accurate Monitoring

The rising frequency of extreme weather events necessitates enhanced monitoring technologies.

Project Scope and Deliverables

Real-time IoT-based weather anomaly detection system

IoT-based Weather Monitoring System

Designed a system using BeagleBone Black and DHT22 sensor to collect real-time temperature and humidity data.

Efficient Data Collection Process

Implemented AWS IoT and DynamoDB for secure and scalable data collection, storage, and real-time data management.

Anomaly Detection and Prediction

Developed machine learning models (SARIMA and LSTM) to predict and identify temperature and humidity anomalies.

Historical Data Integration

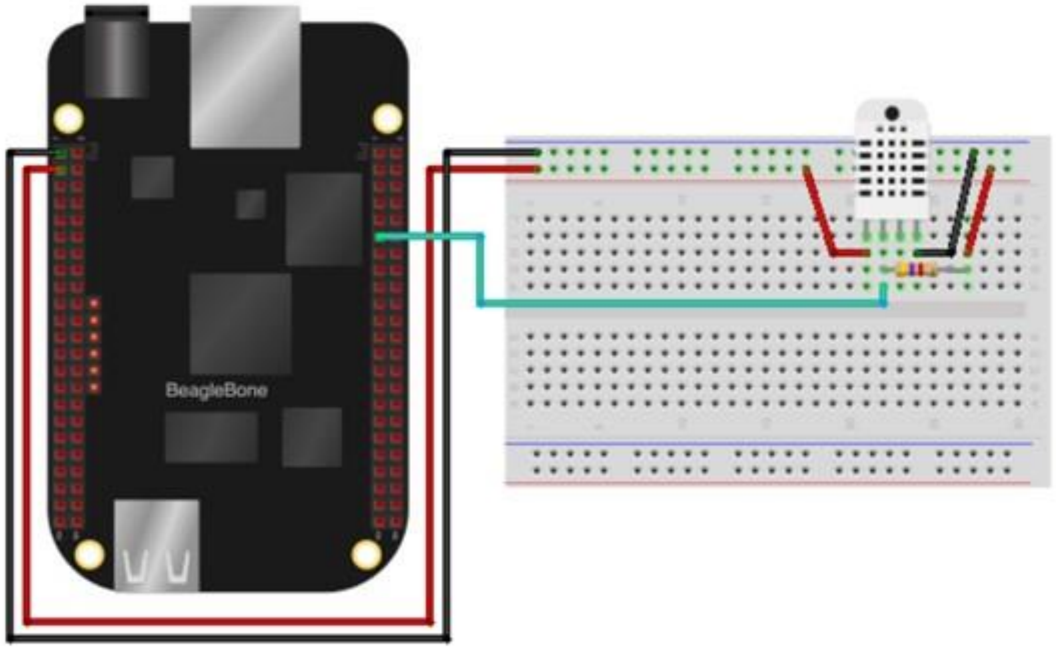
Integrated Meteostat API for retrieving historical weather data and stored in MongoDB to enhance machine learning model predictions.

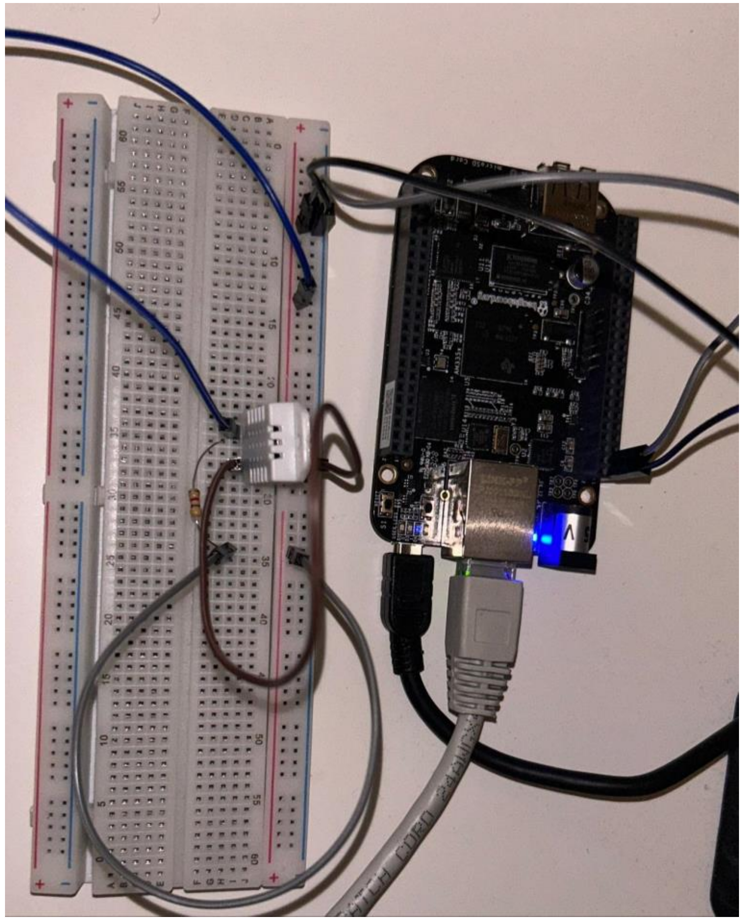
Data Visualization

Created a user-friendly interface displaying real-time weather data, anomalies, and future forecasts using plots and charts.

Hardware Setup

Utilizing BeagleBone Black and DHT22 Sensors for Temperature & Humidity Data Collection





Software Setup

Integration of IoT, Cloud, and Machine Learning Technologies

1

Python Programming Language

The project is developed using Python, which is ideal for data processing, machine learning, and interfacing with hardware (BeagleBone Black) and cloud services (AWS IoT and DynamoDB).

2

Libraries and Tools

Key Python libraries used include Boto3, Pandas and NumPy, Matplotlib, Statsmodels, PySpark, PyTorch

3

Cloud Services

AWS IoT and DynamoDB were integrated for real-time data transmission and storage. MongoDB was used to store historical weather data fetched from the Meteostat API for model training and predictions.

4

Real-Time Data Processing

The system leverages PySpark to handle real-time data streams, enabling anomaly detection and real-time data analysis at scale.

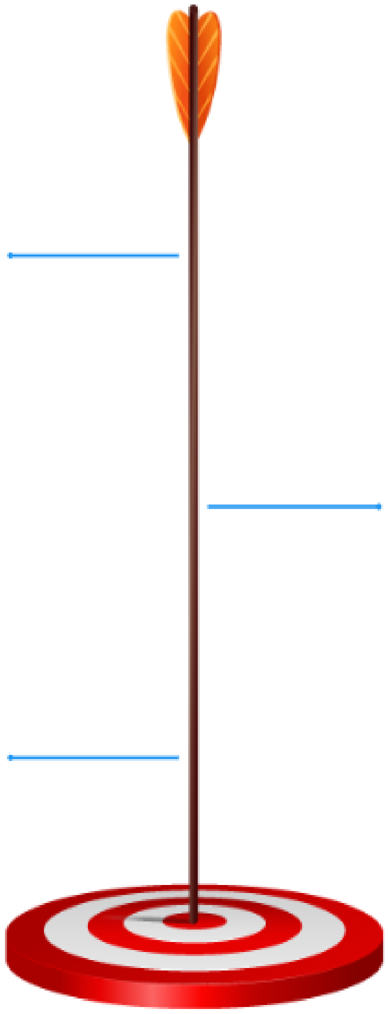
5

Machine Learning Models

Two machine learning models, SARIMA and LSTM, are used to detect anomalies in both real-time and forecasted weather data. The SARIMA model forecasts temperature trends using historical data, while LSTM enhances prediction accuracy through sequence learning.

Cloud Infrastructure Overview

Key Components for Modern Data Management Solutions



AWS IoT

Enables secure data transmission for connected devices, ensuring integrity.

AWS DynamoDB

Provides real-time data storage, supporting high-speed data retrieval and scalability.

MongoDB

Stores historical data, facilitating machine learning and advanced analytics.

Cloud Infrastructure Overview

Key Components for Modern Data Management Solutions

Documents 32.0K | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

ADD DATA | EXPORT DATA | UPDATE | DELETE

snow : NaN
wdir : 113
wspd : 8
wpgt : 25.9
pres : 1009.5
tsun : NaN

_id: ObjectId('66d861bb154b8b5fa499d735')
time : 2019-01-04T00:00:00.000+00:00
tavg : 3.3
tmin : -0.2
tmax : 6
prcp : 2.8
snow : NaN
wdir : 231
wspd : 20.5
wpgt : 51.8
pres : 1013.1
tsun : NaN

_id: ObjectId('66d861bb154b8b5fa499d736')
time : 2019-01-05T00:00:00.000+00:00
tavg : 0.4
tmin : -1
tmax : 2
prcp : 0.3
snow : NaN
wdir : 5
wspd : 8.6
wpgt : 31.5
pres : 1014.5
tsun : NaN

sensor_id *(String)*	timestamp *(String)*	humidity	temperature
sensor1	2024-08-30 14:46:26	60	25.6
sensor1	2024-08-30 14:46:37	60	25.6
sensor1	2024-08-30 14:46:47	60	25.6
sensor1	2024-08-30 14:46:57	60	25.6
sensor1	2024-08-30 15:08:34	36	30
sensor1	2024-08-30 15:08:44	36	30
sensor1	2024-08-30 15:08:54	36	30
sensor1	2024-08-31 18:55:09	41.309968...	26.960529539243165
sensor1	2024-08-31 18:55:19	36.457175...	28.07909155206312
sensor1	2024-08-31 18:55:29	36.186274...	28.376410740896965
sensor1	2024-08-31 18:55:39	39.643250...	26.50942380587486
sensor1	2024-08-31 18:55:49	42.282792...	28.48840208302055
sensor1	2024-08-31 18:56:00	43.259516...	27.73218299117893
sensor1	2024-08-31 18:56:10	40.909060...	28.13407704039244
sensor1	2024-08-31 18:56:20	35.731592...	28.548566667504936

System Architecture

Overview of the System Components

Data Collection

Utilizing DHT22 sensor for environment data acquisition, connected to the BeagleBone Black and historical data from meteostat API.

Real-Time Data Transmission

Sending the collected data in real-time to AWS DynamoDB for secure storage and real-time access.

Anomaly Detection

Applying predefined thresholds to detect anomalies in the collected data, such as unusual temperature or humidity variations.

Machine Learning

Implementing SARIMA ,LSTM and hyrid model for predicting future trends based on historical data.

Real-Time Data Collection Algorithm

An overview of the essential steps in data collection and transmission

Initialization Phase

Initialize sensor and AWS session

Commence the process by setting up the necessary sensors and establishing a session with AWS, ensuring a secure connection for data handling.

Data Reading Interval

Read data every 10 seconds

Implement a routine that captures data from the sensors at regular intervals of 10 seconds, maintaining a consistent flow of information.

Data Transmission

Transmit valid data to AWS DynamoDB

Once valid data is collected, transmit it efficiently to AWS DynamoDB for storage and further processing, ensuring data integrity and availability.

Error Handling

Handle errors and retry

Incorporate error-handling mechanisms to address any issues during data collection or transmission.

SARIMA

SARIMA Use Cases and My Application

SARIMA is useful in domains where time series data exhibits seasonality:

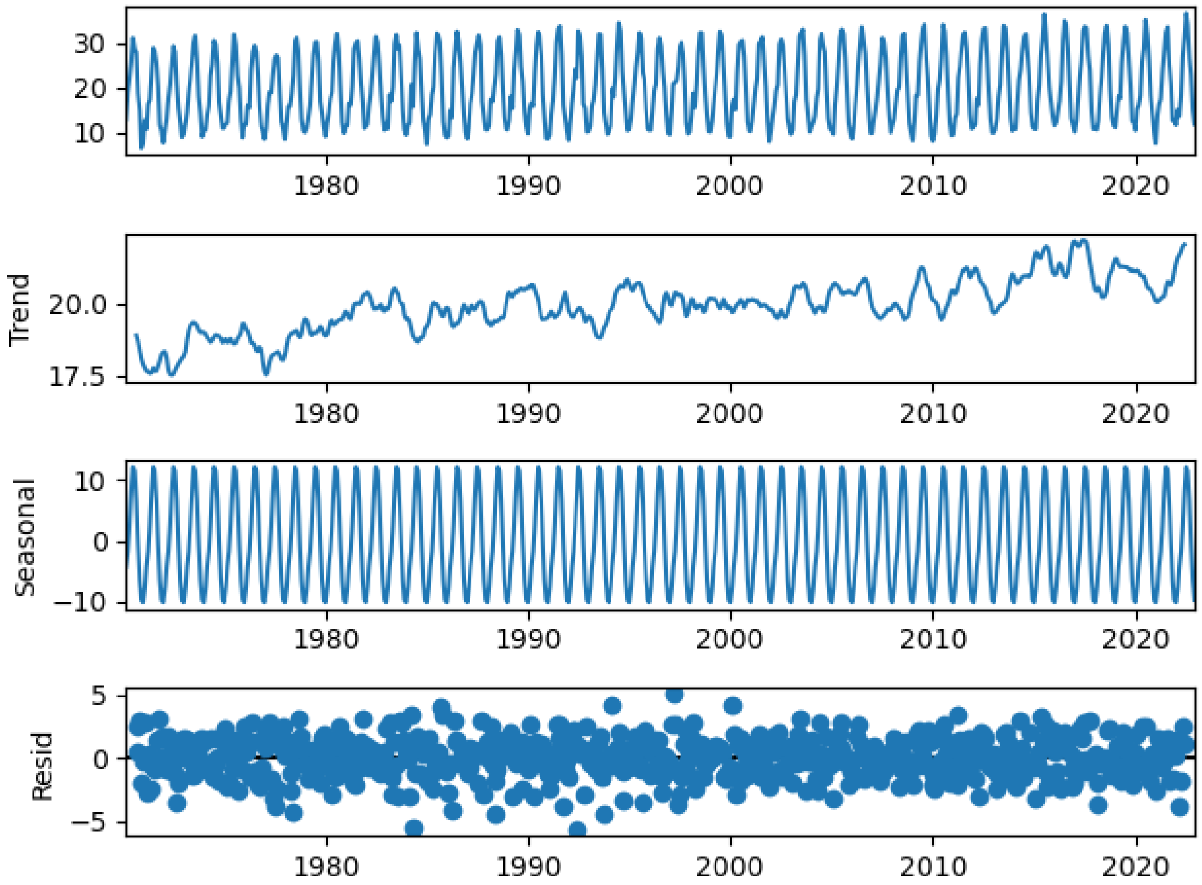
- **Weather forecasting:** Captures temperature patterns over weeks and months.
- **Sales forecasting:** Tracks monthly or quarterly trends in sales.
- **Energy demand prediction:** For electricity consumption with daily/weekly cycles.

In my case: SARIMA was applied to predict daily temperatures. Since temperature data has both weekly and monthly cycles, I selected:

- **Weekly period:** $m = 7$
- **Monthly period:** $m = 12$

Why SARIMA?

- Captures both linear trends and seasonality.
- Easy to interpret and widely used in time series forecasting.



SARIMA

SARIMA is an extension of the ARIMA (Autoregressive Integrated Moving Average) model, designed to handle time series data that has both trends and seasonal patterns.

ARIMA Formula:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \tag{1}$$

Where:

- y_t: Value at time t
- ϕ: AR coefficients (past values)
- θ: MA coefficients (past errors)
- ϵ_t: Error term (white noise)

```
# Train SARIMA Model
sarima_order = (1, 1, 1)
seasonal_order = (1, 1, 1, 12)
sarima_model = SARIMAX(df_mongo['tavg'], order=sarima_order, seasonal_order=seasonal_order)
sarima_model_fit = sarima_model.fit(disp=False)
```

SARIMA

In SARIMA, we add seasonal components (e.g., weekly, monthly) to capture repeating patterns:

$$(1 - \Phi B^s)(1 - \phi B)(1 - B)^d y_t = (1 + \Theta B^s)(1 + \theta B)\epsilon_t \tag{1}$$

Where:

- B: Backshift operator
- s: Seasonal period (e.g., 7 for weekly, 12 for monthly)
- d: Differencing order (to remove trend)

SARIMA

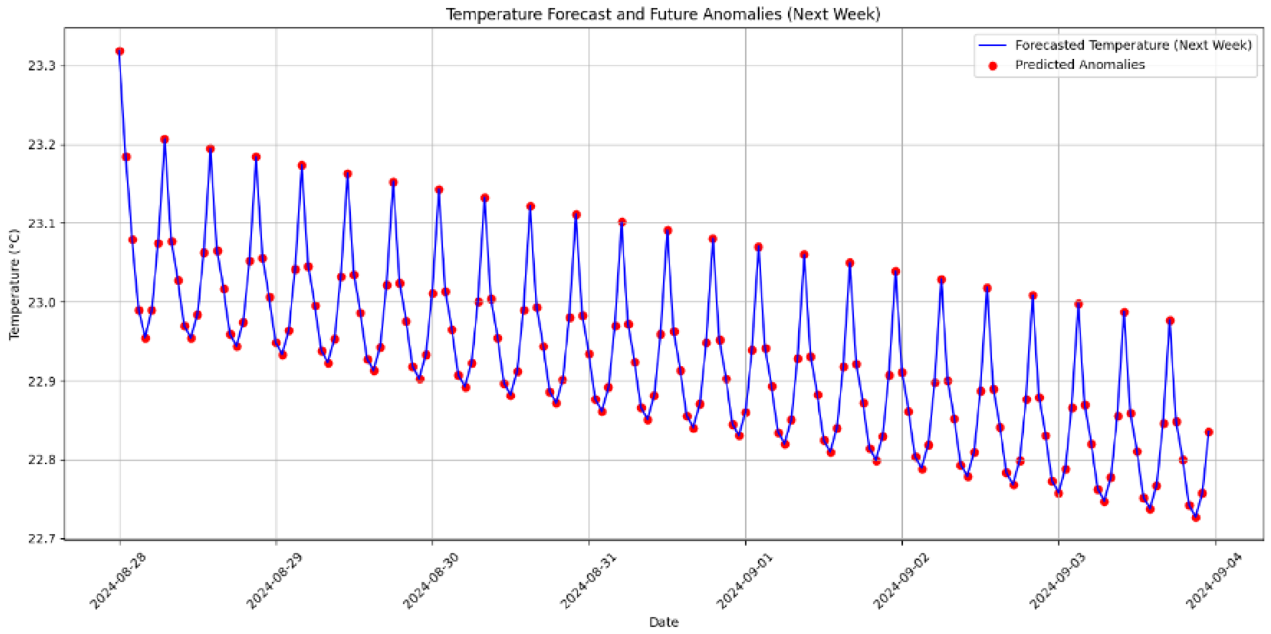
Challenges and Errors in SARIMA

Common Issues:

- **Overfitting:** When too many parameters are included, the model fits the training data too well but performs poorly on new data.
- **Underfitting:** When the model is too simple and fails to capture important patterns.
- **Seasonal Misidentification:** Selecting the wrong seasonal parameters can lead to inaccurate predictions.
- **Stationarity:** SARIMA assumes the data is stationary, meaning constant mean and variance. Achieving this requires proper differencing.

Challenges in My Case:

- Predicting non-linear temperature fluctuations during extreme weather events.
- SARIMA struggled with real-time anomaly detection.



Algorithms Overview

Exploring Key Algorithms for Data Management and Anomaly Detection

Algorithm 2 Real-Time Anomaly Detection

Fetch sensor data from AWS DynamoDB.
Define thresholds for temperature and humidity (e.g., 40°C and 70%).
Compare real-time data against thresholds.
if anomaly detected **then**
    Display anomaly alert in GUI.
else
    Display normal status in GUI.
end if
Provide a refresh option for updating the displayed data.

Algorithm 3 SARIMA Model Training and Anomaly Prediction

Fetch and preprocess historical weather data from MongoDB.
Train the SARIMA model using the preprocessed data.
Define thresholds for future temperature anomalies (e.g., 35°C).
Use the SARIMA model to predict temperature for the next 7 days.
Compare predicted temperatures against anomaly thresholds.
if anomalies predicted **then**
    Flag the predicted anomalies for the alert.
end if
Visualize the predictions and anomalies.

Algorithms Overview

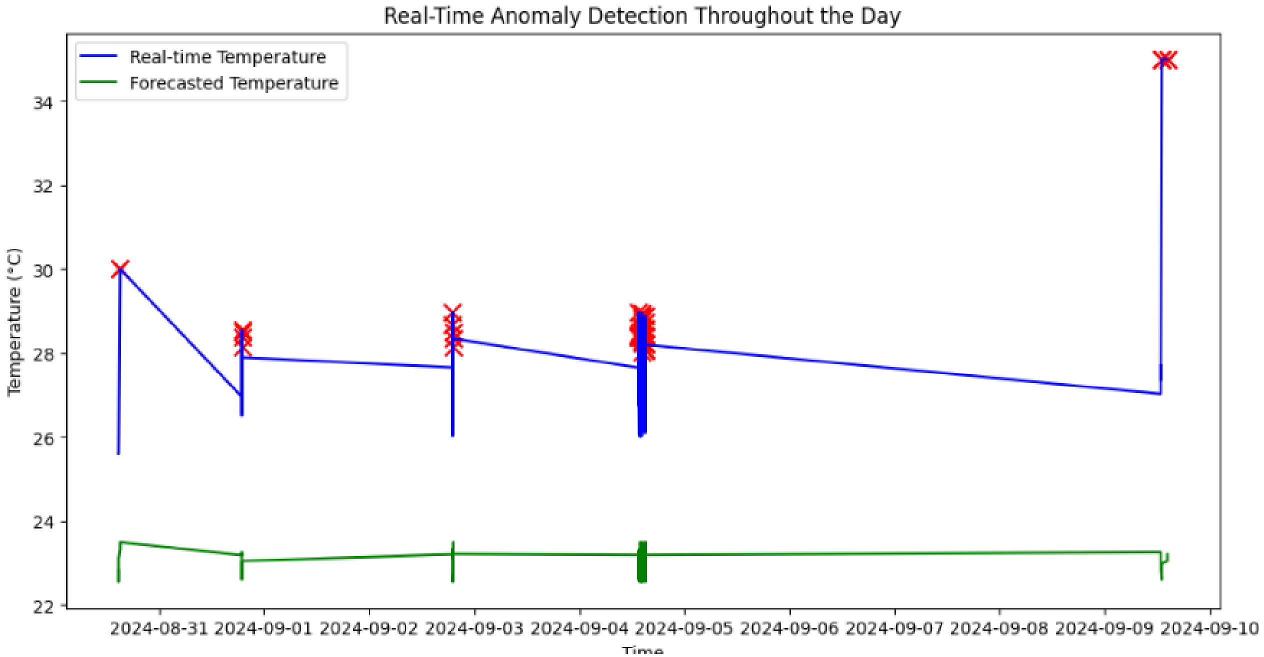
Exploring Key Algorithms for Data Management and Anomaly Detection

Algorithm 4 Real-Time Anomaly Detection and Forecasting Workflow

```
while True do
    Fetch and display real-time data and anomalies.
    Predict future anomalies using the SARIMA model.
    Visualize real-time and future data.
    Sleep for 60 seconds before next real-time data update.
end while
```

Algorithms Overview

Exploring Key Algorithms for Data Management and Anomaly Detection




```
Timestamp: 2024-08-30 14:46:26, Current Temp: 25.6, Forecast Temp: 22.83436005376649, Anomaly: False
Timestamp: 2024-08-30 14:46:37, Current Temp: 25.6, Forecast Temp: 22.55434221003204, Anomaly: False
Timestamp: 2024-08-30 14:46:47, Current Temp: 25.6, Forecast Temp: 22.775900987078064, Anomaly: False
Timestamp: 2024-08-30 14:46:57, Current Temp: 25.6, Forecast Temp: 23.11195619760667, Anomaly: False
Timestamp: 2024-08-30 15:08:34, Current Temp: 30.0, Forecast Temp: 23.324653539790027, Anomaly: True
Timestamp: 2024-08-30 15:08:44, Current Temp: 30.0, Forecast Temp: 23.433761463282096, Anomaly: True
Timestamp: 2024-08-30 15:08:54, Current Temp: 30.0, Forecast Temp: 23.49809548519086, Anomaly: True
Timestamp: 2024-08-31 18:55:09, Current Temp: 26.960529539243165, Forecast Temp: 23.186876973473762, Anomaly: False
Timestamp: 2024-08-31 18:55:19, Current Temp: 28.07909155206312, Forecast Temp: 23.21766625642965, Anomaly: False
Timestamp: 2024-08-31 18:55:29, Current Temp: 28.376410740896965, Forecast Temp: 23.192424574720707, Anomaly: True
Timestamp: 2024-08-31 18:55:39, Current Temp: 26.50942380587486, Forecast Temp: 23.259603983723558, Anomaly: False
Timestamp: 2024-08-31 18:55:49, Current Temp: 28.48840208302055, Forecast Temp: 23.15750901754837, Anomaly: True
Timestamp: 2024-08-31 18:56:00, Current Temp: 27.73218299117893, Forecast Temp: 22.832353678677645, Anomaly: False
Timestamp: 2024-08-31 18:56:10, Current Temp: 28.13407704039244, Forecast Temp: 22.60534687555322, Anomaly: True
Timestamp: 2024-08-31 18:56:20, Current Temp: 28.548566667504936, Forecast Temp: 22.75258510397249, Anomaly: True
Timestamp: 2024-08-31 18:56:30, Current Temp: 27.56511754976714, Forecast Temp: 22.985068938522012, Anomaly: False
Timestamp: 2024-08-31 18:56:40, Current Temp: 27.89690966011864, Forecast Temp: 23.049207741107207, Anomaly: False
Timestamp: 2024-09-02 19:03:49, Current Temp: 27.657508808961367, Forecast Temp: 23.209982544235828, Anomaly: False
Timestamp: 2024-09-02 19:03:59, Current Temp: 26.32964897654297, Forecast Temp: 23.291206178371652, Anomaly: False
Timestamp: 2024-09-02 19:04:09, Current Temp: 26.02720798598098, Forecast Temp: 23.123513851412845, Anomaly: False
Timestamp: 2024-09-02 19:04:19, Current Temp: 26.98062072455593, Forecast Temp: 23.2440108520488, Anomaly: False
Timestamp: 2024-09-02 19:04:29, Current Temp: 27.85068000758714, Forecast Temp: 23.27335327841281, Anomaly: False
Timestamp: 2024-09-02 19:04:40, Current Temp: 26.96390011514821, Forecast Temp: 23.329714801982433, Anomaly: False
Timestamp: 2024-09-02 19:04:50, Current Temp: 28.968941670191427, Forecast Temp: 23.199644571542485, Anomaly: True
Timestamp: 2024-09-02 19:05:00, Current Temp: 27.192504029820167, Forecast Temp: 22.83436005376649, Anomaly: False
Timestamp: 2024-09-02 19:05:10, Current Temp: 28.68424353604756, Forecast Temp: 22.55434221003204, Anomaly: True
Timestamp: 2024-09-02 19:11:14, Current Temp: 26.84390259771385, Forecast Temp: 22.775900987078064, Anomaly: False
Timestamp: 2024-09-02 19:11:24, Current Temp: 28.137170792246415, Forecast Temp: 23.11195619760667, Anomaly: True
Timestamp: 2024-09-02 19:11:34, Current Temp: 26.605295899863112, Forecast Temp: 23.324653539790027, Anomaly: False
Timestamp: 2024-09-02 19:11:44, Current Temp: 26.24625242520393, Forecast Temp: 23.433761463282096, Anomaly: False
Timestamp: 2024-09-02 19:11:54, Current Temp: 28.503031560438476, Forecast Temp: 23.49809548519086, Anomaly: True
Timestamp: 2024-09-02 19:12:04, Current Temp: 26.97754681848841, Forecast Temp: 23.186876973473762, Anomaly: False
Timestamp: 2024-09-02 19:12:15, Current Temp: 28.34452432027105, Forecast Temp: 23.21766625642965, Anomaly: True
Timestamp: 2024-09-04 13:30:00, Current Temp: 27.65262732002807, Forecast Temp: 23.192424574720707, Anomaly: False
Timestamp: 2024-09-04 13:30:11, Current Temp: 26.76400489591873, Forecast Temp: 23.259603983723558, Anomaly: False
Timestamp: 2024-09-04 13:30:21, Current Temp: 27.787167069179326, Forecast Temp: 23.15750901754837, Anomaly: False
Timestamp: 2024-09-04 13:30:31, Current Temp: 28.37210575605275, Forecast Temp: 22.832353678677645, Anomaly: True
Timestamp: 2024-09-04 13:30:41, Current Temp: 28.388744270677535, Forecast Temp: 22.60534687555322, Anomaly: True
Timestamp: 2024-09-04 13:30:51, Current Temp: 28.44999483440575, Forecast Temp: 22.75258510397249, Anomaly: True
Timestamp: 2024-09-04 13:31:01, Current Temp: 28.42299518703833, Forecast Temp: 22.985068938522012, Anomaly: True
Timestamp: 2024-09-04 13:31:11, Current Temp: 28.97065329703373, Forecast Temp: 23.049207741107207, Anomaly: True
Timestamp: 2024-09-04 13:31:21, Current Temp: 28.17622173849285, Forecast Temp: 23.209982544235828, Anomaly: False
Timestamp: 2024-09-04 13:31:31, Current Temp: 27.915669721902198, Forecast Temp: 23.291206178371652, Anomaly: False
Timestamp: 2024-09-04 13:43:18, Current Temp: 26.06412038004539, Forecast Temp: 23.123513851412845, Anomaly: False
```

SARIMA - LSTM Hybrid Model

Approach and Logic

SARIMA captures linear and seasonal components, but struggles with non-linear relationships. I combined SARIMA with Long Short-Term Memory (LSTM) networks to handle non-linear patterns.

Why LSTM?

- **Long-term dependencies:** LSTM captures long-term dependencies, ideal for non-linear trends.
- **Residual Learning:** LSTM refines predictions by learning from SARIMA's forecast errors.

Hybrid Model Formula:

$$\hat{y}_t = y_{\hat{SARIMA}} + y_{\hat{LSTM}} \quad (2$$

Where:

- $y_{\hat{SARIMA}}$ is SARIMA's forecast.
- $y_{\hat{LSTM}}$ is LSTM's forecasted residual (error).

```
def create_sequences(data, seq_length):
    sequences = []
    labels = []
    for i in range(len(data) - seq_length):
        sequences.append(data[i:i+seq_length])
        labels.append(data[i+seq_length])
    return np.array(sequences), np.array(labels)

seq_length = 30
X, y = create_sequences(scaled_residuals, seq_length)

X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32)


class LSTMModel(nn.Module):
    def __init__(self, input_size=1, hidden_layer_size=100, output_size=1):
        super(LSTMModel, self).__init__()
        self.hidden_layer_size = hidden_layer_size
        self.lstm = nn.LSTM(input_size, hidden_layer_size, batch_first=True)
        self.linear = nn.Linear(hidden_layer_size, output_size)

    def forward(self, input_seq):
        lstm_out, _ = self.lstm(input_seq)
        predictions = self.linear(lstm_out[:, -1, :])
        return predictions

model = LSTMModel()
loss_function = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

SARIMA - LSTM Hybrid Model

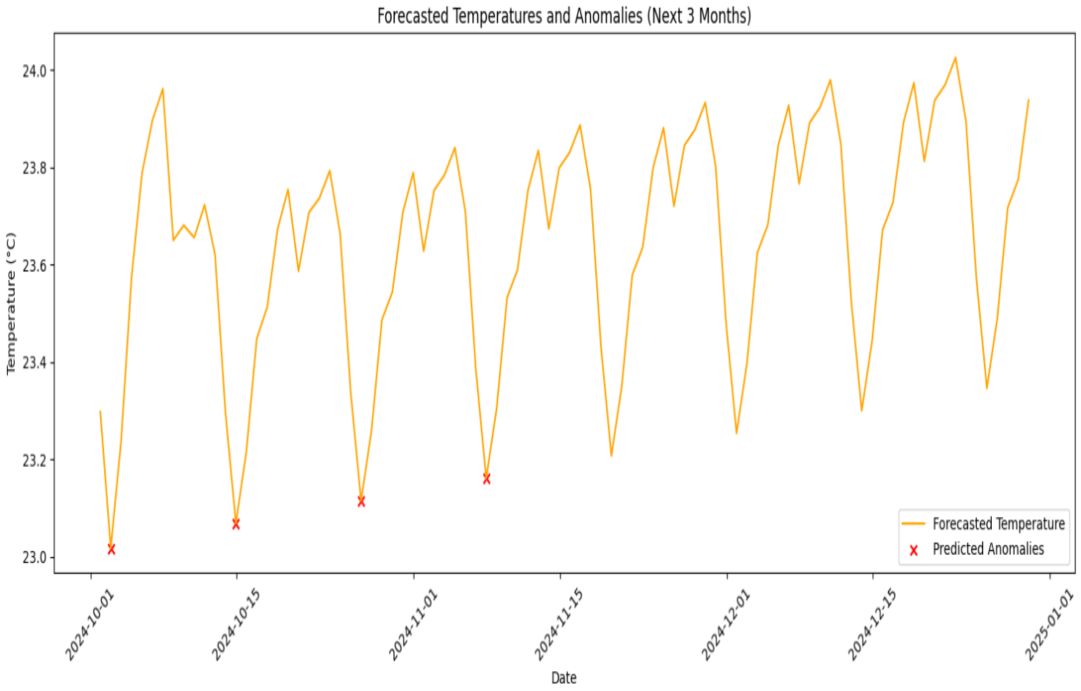
Why Hybrid Approach: My Use Case

Motivation for Hybrid Model:

- SARIMA alone failed to capture the non-linear complexities in temperature fluctuations.
- LSTM modeled residuals, capturing non-linear behavior during extreme weather.

Implementation:

- **Step 1:** SARIMA forecasted the next 90 days of temperature.
- **Step 2:** LSTM learned the non-linear residuals from SARIMA's predictions.
- **Step 3:** The final prediction combined outputs from both models, improving forecast accuracy.



SARIMA - LSTM Hybrid Model

Real-Time Data Integration and Anomaly Detection

Real-Time Data Integration:

- I used IoT sensor data (temperature, humidity) for continuous model updates.

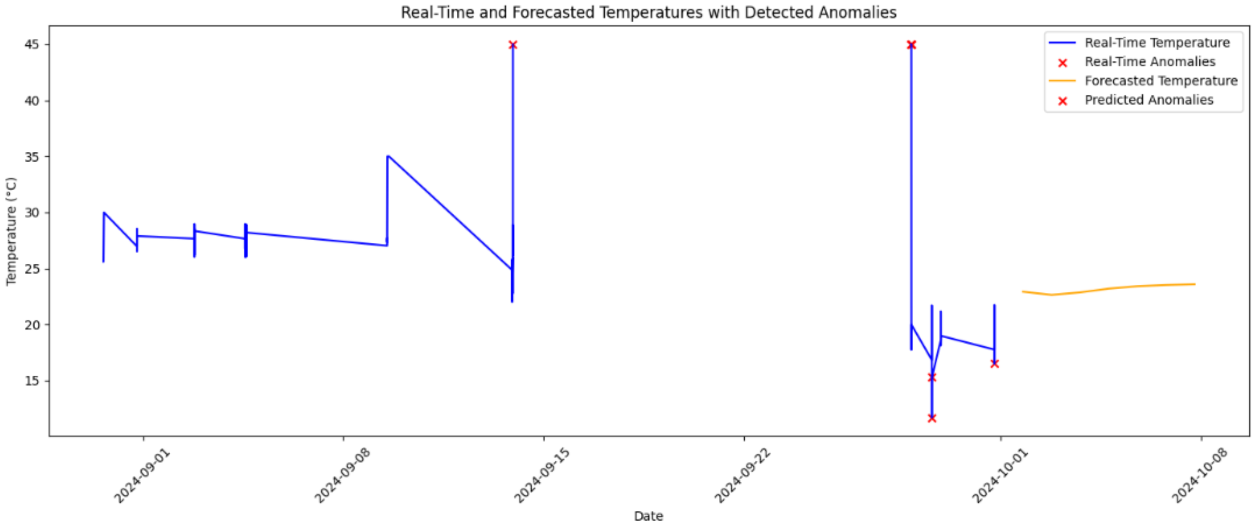
Anomaly Detection Formula:

$$anomaly = |y_t - \hat{y}_t| > threshold \quad (3)$$

Where:

- y_t: Real-time temperature
- $\hat{y}_t$: Forecasted temperature (SARIMA + LSTM)
- The threshold is based on historical variance.

Real-time anomaly detection is critical for applications such as weather anomaly forecasting.




```python
def detect_anomalies(temperature_values, predicted_values=None, threshold_factor=2):
    """Detect anomalies based on the temperature deviations from the mean or predicted values."""
    if predicted_values is None:
        mean_temp = np.mean(temperature_values)
        std_temp = np.std(temperature_values)
        upper_threshold = mean_temp + threshold_factor * std_temp
        lower_threshold = mean_temp - threshold_factor * std_temp
        anomalies = (temperature_values > upper_threshold) | (temperature_values < lower_threshold)
    else:
        mean_temp = np.mean(predicted_values)
        std_temp = np.std(predicted_values)
        upper_threshold = mean_temp + threshold_factor * std_temp
        lower_threshold = mean_temp - threshold_factor * std_temp
        anomalies = (temperature_values > upper_threshold) | (temperature_values < lower_threshold)
    return anomalies
```

Algorithms Overview

Exploring Key Algorithms for Data Management and Anomaly Detection

Algorithm 5 Hybrid SARIMA-LSTM Approach for Anomaly Detection

1: Fetch historical weather data from MongoDB, ensuring daily frequency and data cleanliness.
2: Train the SARIMA model to predict future temperature values based on seasonal patterns.
3: Calculate residuals by subtracting SARIMA predictions from actual observed temperatures.
4: Scale residuals and create sequences for input into the LSTM model.
5: Train the LSTM model on the scaled residuals to predict future deviations from SARIMA forecasts.
6: Fetch real-time weather data from AWS DynamoDB for anomaly detection.
7: Detect anomalies based on temperature deviations from the SARIMA and LSTM predictions using a threshold-based mechanism.
8: Visualize real-time data, forecasted values, residuals, and anomalies for continuous monitoring and analysis.
9: Continuously process real-time data, update predictions, and detect anomalies in a real-time pipeline.

SARIMA - LSTM Hybrid Model

SARIMA vs. Hybrid Model: Comparison

SARIMA Pros:

- Simple, interpretable, and effective for regular patterns.
- Well-suited for linear time series with clear seasonality.

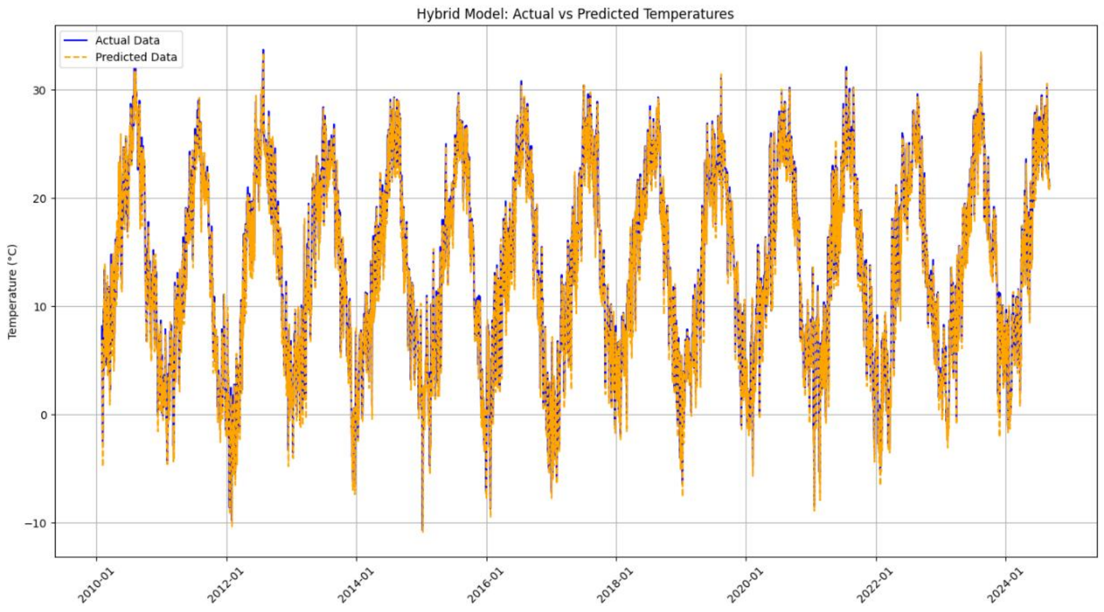
Hybrid Model Pros:

- Captures both linear and non-linear patterns.
- Better suited for real-world, complex scenarios with irregular patterns.

Performance in My Case:

- **SARIMA:** Worked well for regular patterns but struggled with anomalies.
- **Hybrid Model:** Improved accuracy by handling both linear and non-linear patterns, especially during weather anomalies.

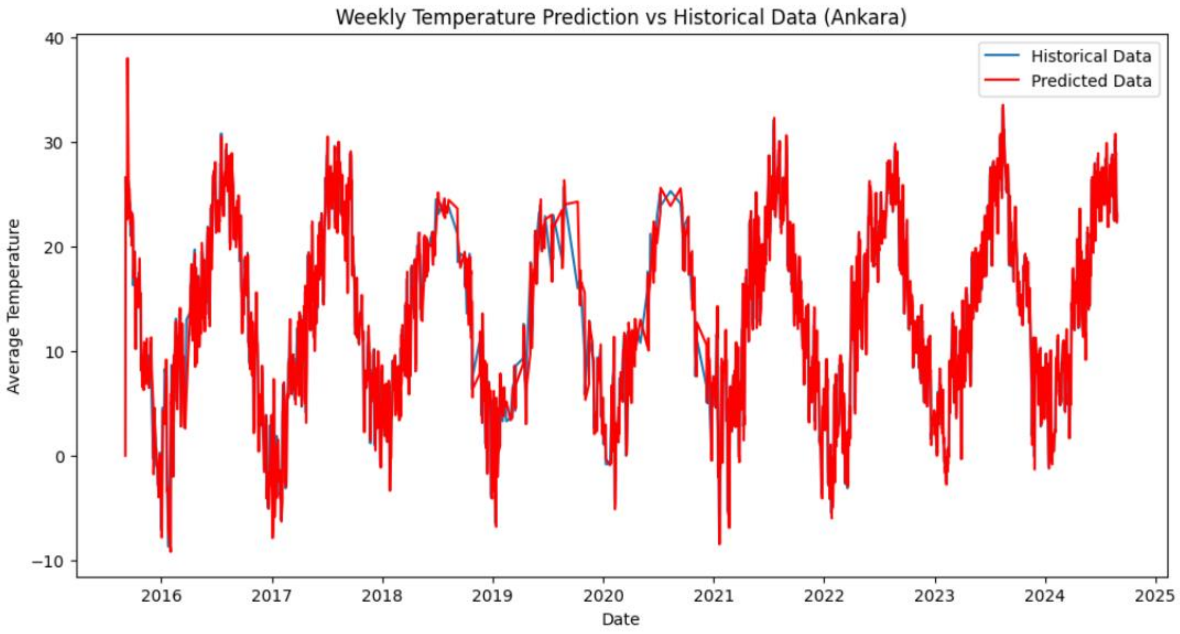
```
MSE: 4.175257428552416
MAE: 1.57594813937161
```



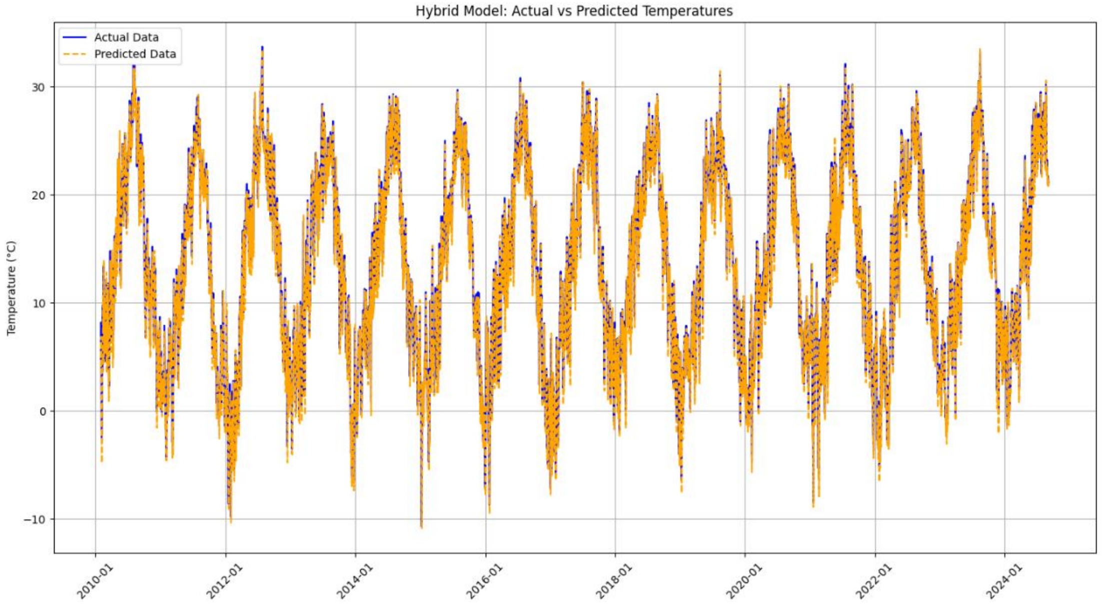
SARIMA - LSTM Hybrid Model

SARIMA vs. Hybrid Model: Comparison

```
Mean Squared Error (MSE): 5.270060926590753
Mean Absolute Error (MAE): 1.6793725942085258
```

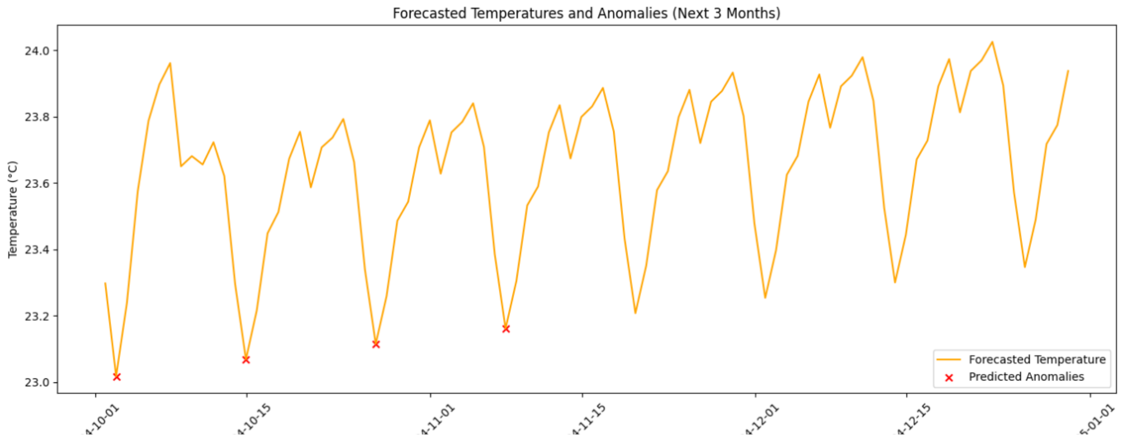


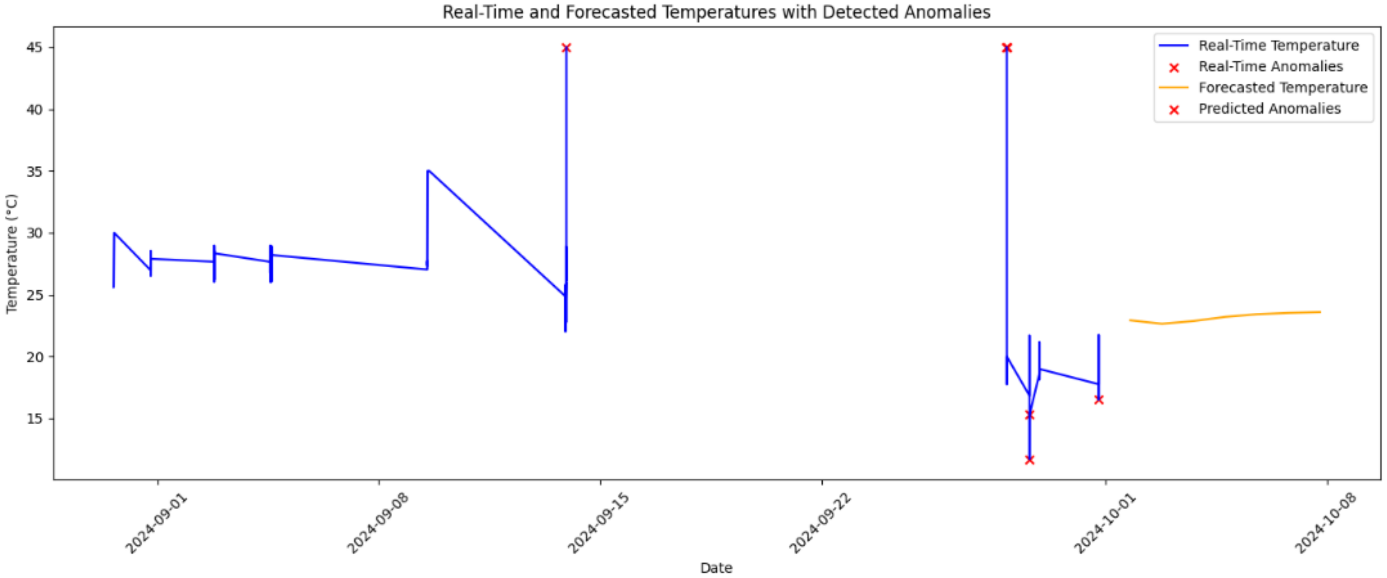

```
MSE: 4.175257428552416
MAE: 1.57594813937161
```



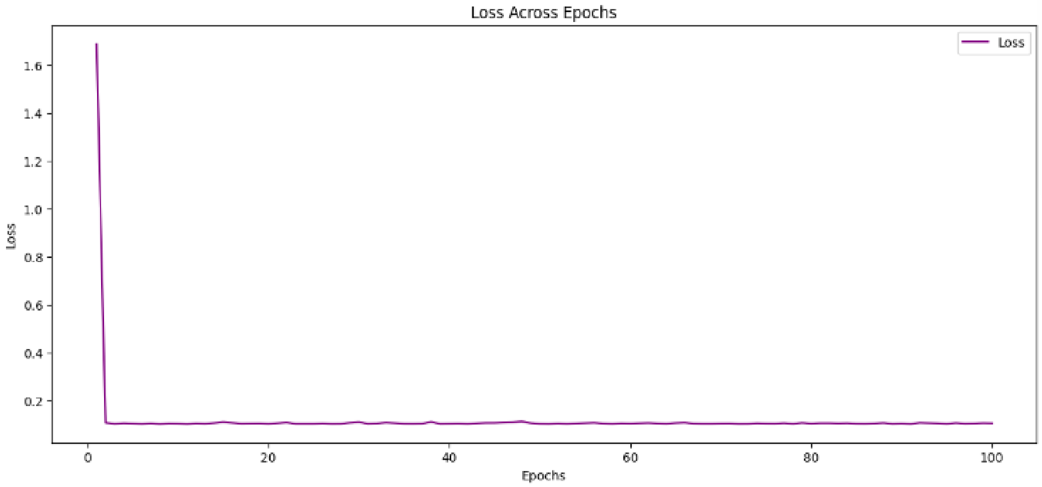
Result and Discussion

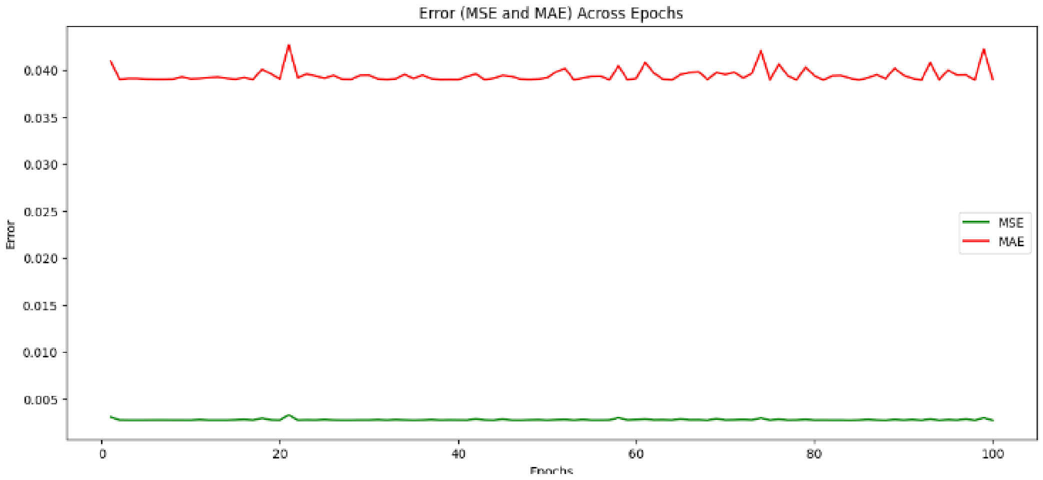
Hybrid Model





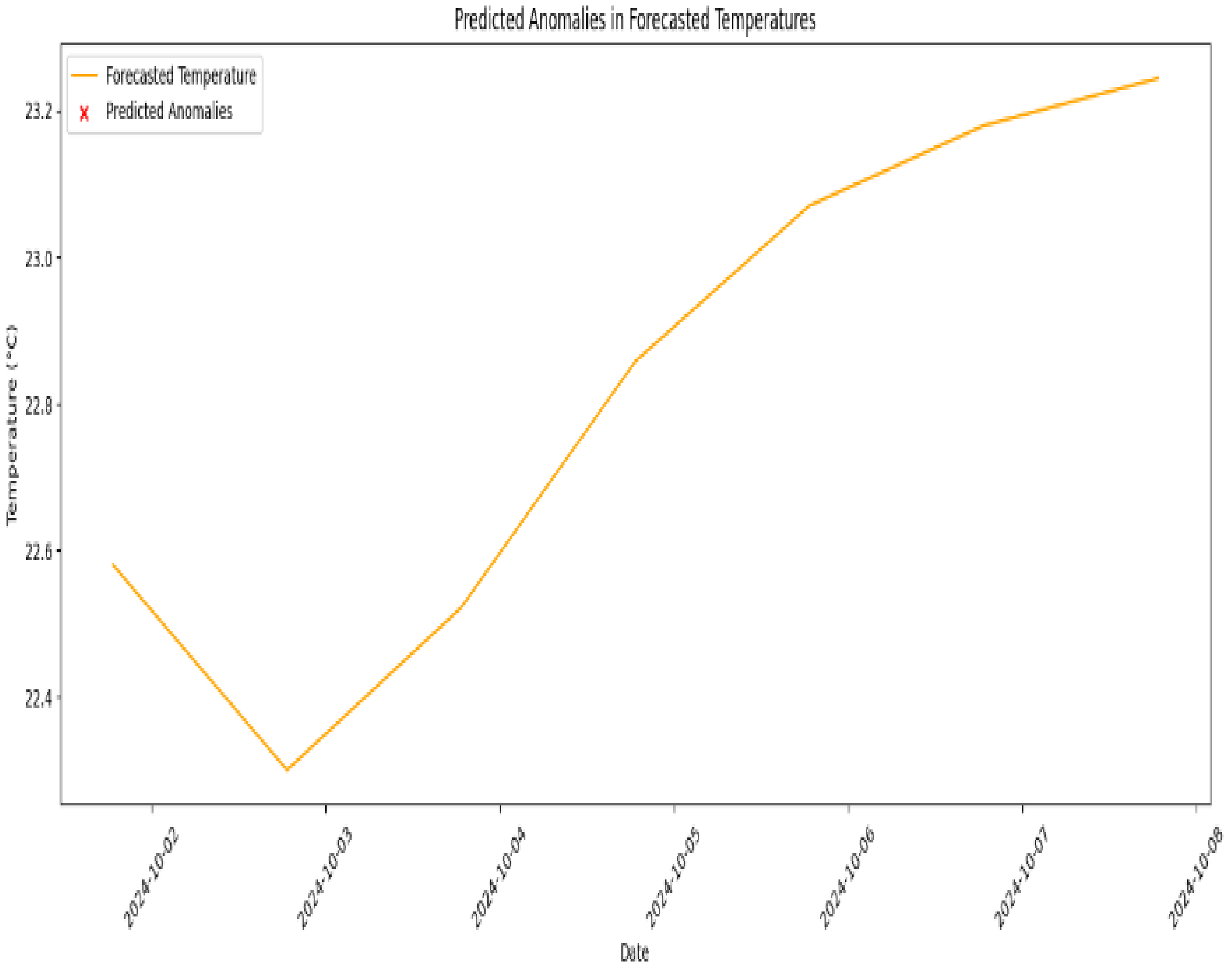
Epoch 100/100, Loss: 0.10757036705035716

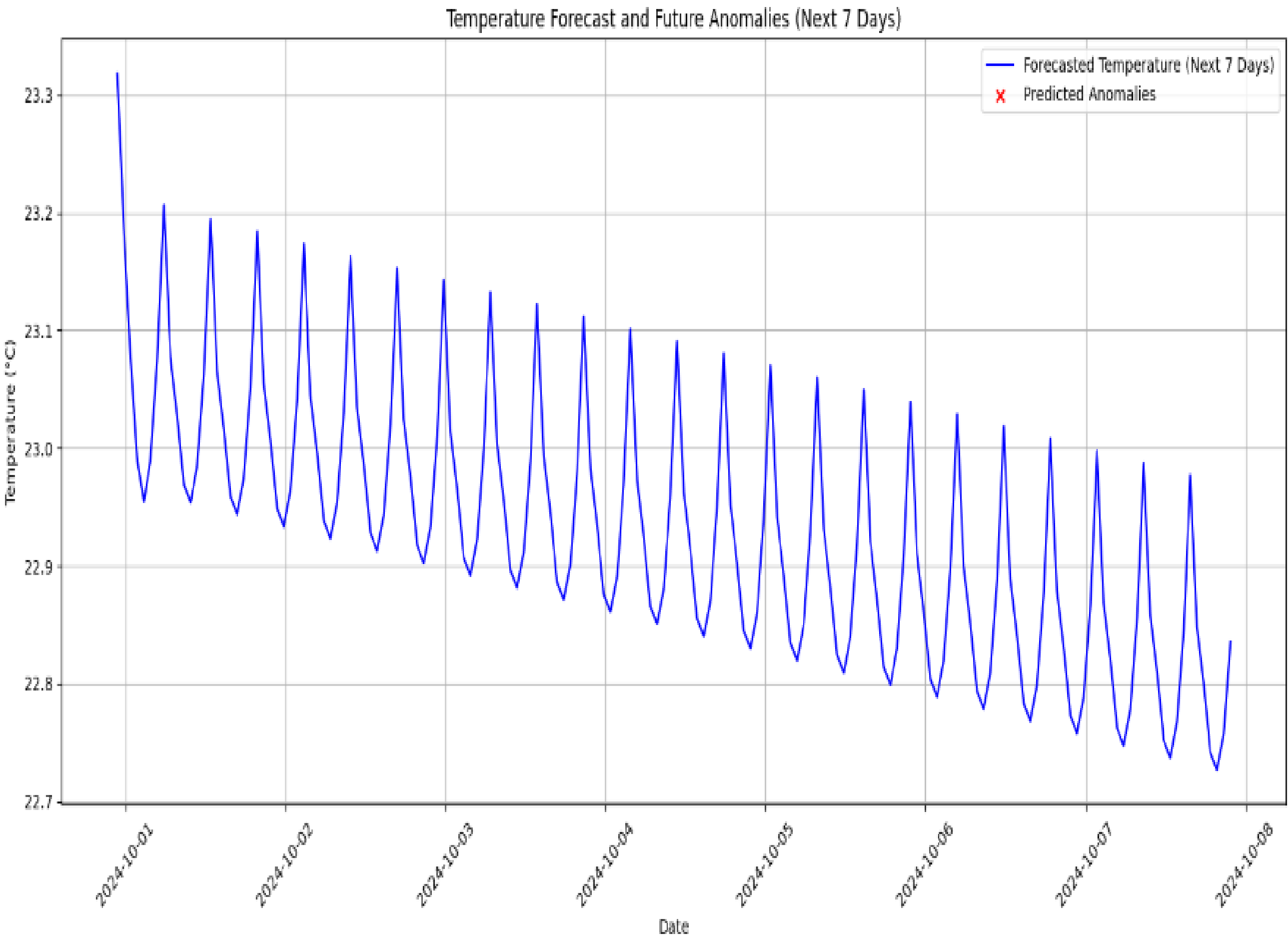




Result and Discussion

Hybrid Model vs SARIMA





Visualizing Weather Trends

Climate Patterns and Data Representation

1. **Understanding Weather Variability**

   Explore the fluctuations in weather patterns over time, highlighting significant changes.

2. **Importance of Data Visualization**

   Using visual tools to make complex weather data accessible and understandable.

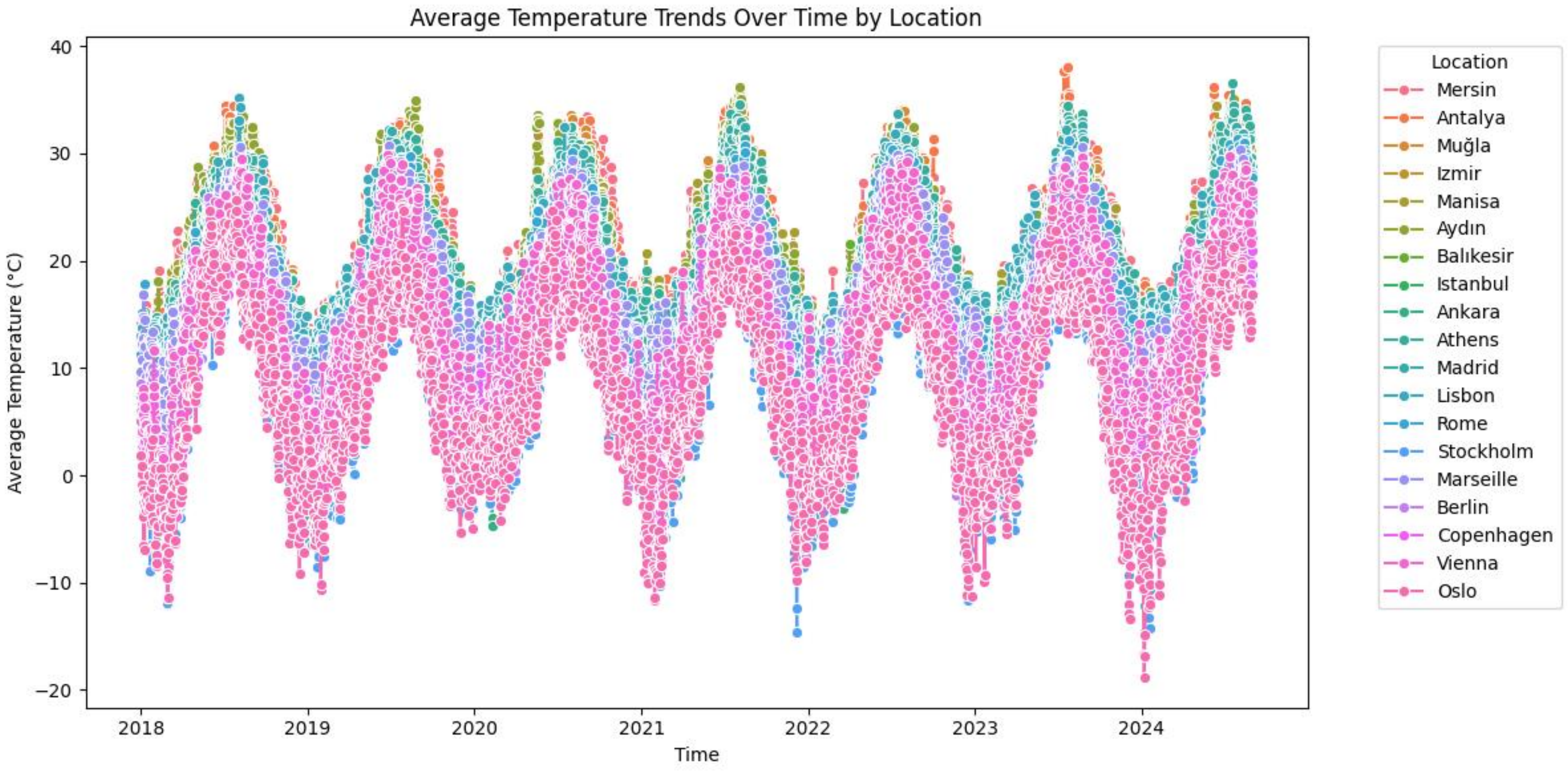
3. **Trends in Temperature Changes**

   Analyze historical temperature data to identify long-term warming trends and anomalies.

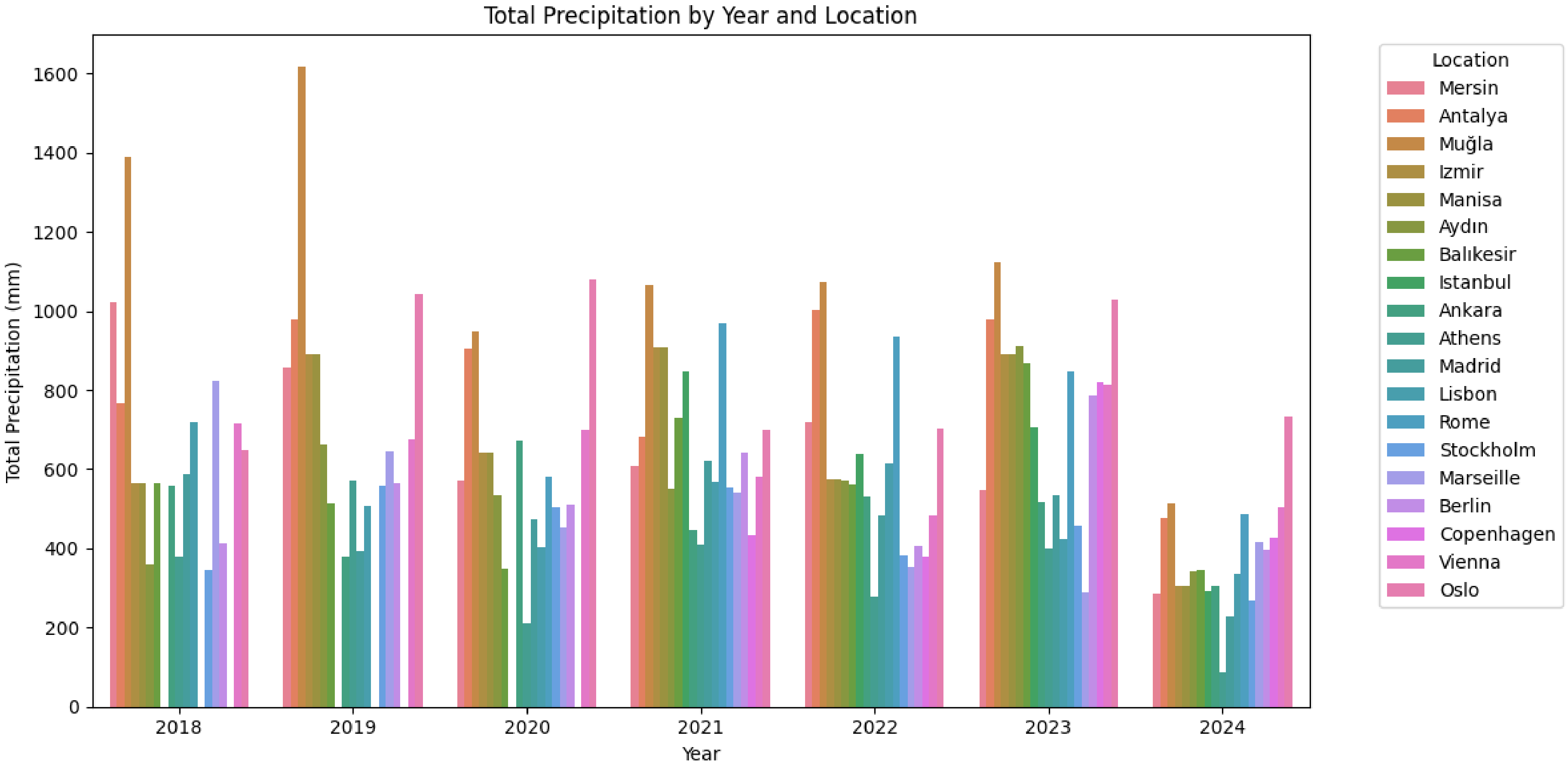
4. **Precipitation Patterns**

   Examine shifts in rainfall patterns and their implications for climate and agriculture.

Weather Analysis

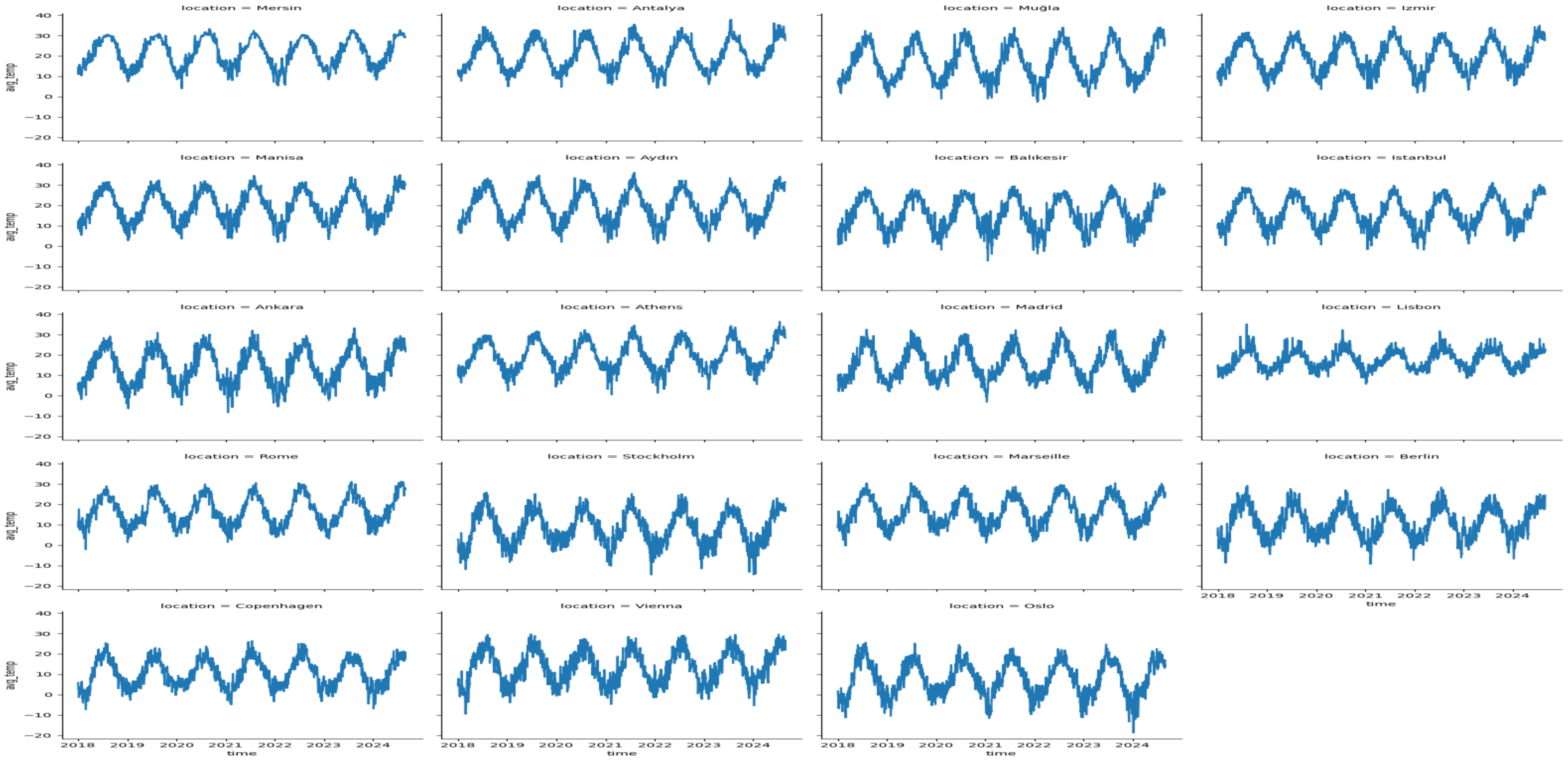


Weather Analysis

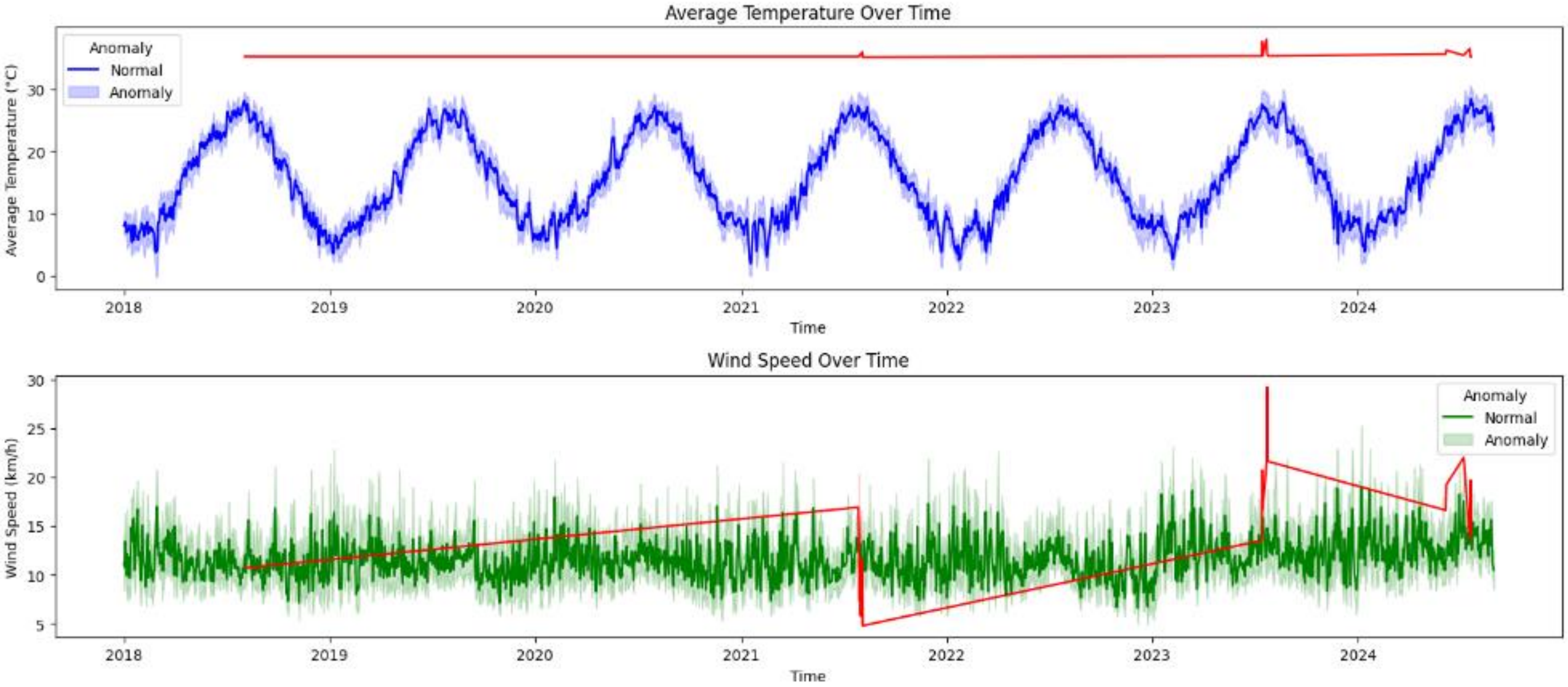


Weather Analysis

Temperature Trends Over Time for Different Locations



Weather Analysis



References

• Meteostat. Available at: https://meteostat.net/en/
• Adafruit Learning System. ”DHT Humidity Sensing on Raspberry Pi with Google Docs Logging”. Available at: https://cdn-learn.adafruit.com/downloads/ pdf/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging.pdf
• G. Coley. ”BeagleBone Black System Reference Manual”, BeagleBoard.org Foundation, 2013. Available at: https://beagleboard.org/black
• A. H. Moghadam and S. F. Shirazi. ”IoT-based smart monitoring and forecasting system for temperature anomalies using cloud technologies”, IEEE Transactions on IoT, vol. 6, pp. 212-221, 2022.
• Hyndman, R. J., & Athanasopoulos, G. (2018). ”Forecasting: principles and practice”. OTexts. Available at: https://otexts.com/fpp3/sarima.html
• Richardson, M. (2015). ”Getting Started with Raspberry Pi”. Maker Media, Inc.
• AWSDocumentation. ”AWS IoT Core- Developer Guide”. Available at: https: //docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html
• Katz, D., Hunger, S. (2017). ”Building the Internet of Things”. Wiley.
• Grolemund, G., Wickham, H. (2017). ”R for Data Science”. O’Reilly Media. Available at: https://r4ds.had.co.nz/
• Scikit-learn Documentation. ”Time Series Forecasting with SARIMA”. Available at: https://scikit-learn.org/stable/modules/linear_model.html#time-series-forecasting
• F. Chollet. ”Deep Learning with Python”. Manning Publications, 2018.
• Documentation for Statsmodels. Available at: https://www.statsmodels.org/ stable/index.html