Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114

Leveraging Python And Apis For Real-Time Weather Data Retrieval And Analysis

[1]Shivam Malani, [2]Salina Parvin, [3]Yash Agrawal, [4]Mohammad Tulaib, [5]Moti Ranjan Tandi
[1,2,3,4]Student of BCA – 6th Semester, [5]Assistant Professor
[1,2,3,4,5]Department of CSIT, Kalinga University, Naya Raipur, Chhattisgarh
[1]Shivammalani17@gmail.com, [2]Salinaali019@gmail.com,
[5]Motiranjan.tandi@kalingauniversity.ac.in

Abstract
This research paper dives into how we can use Python programming along with third-party APIs to gather, analyze, and visualize real-time weather data. As climate change becomes a hot topic in global conversations, having timely and accurate weather information is essential for various sectors like agriculture, transportation, and disaster management. We outline a detailed process for collecting weather data through RESTful APIs, processing it with Python libraries such as Pandas and NumPy, and visualizing trends using tools like Matplotlib and Seaborn. Our system also includes automated testing to guarantee data accuracy and application reliability. By conducting thorough data analysis and creating visualizations, this research underscores the importance and practicality of leveraging Python and APIs for real-time weather insights. With comparative graphs and real-time dashboards, we showcase how well the system can manage dynamic data sources, making this method particularly relevant in the realm of environmental informatics.

Keywords: Python, APIs, Real-Time Weather Data, Data Analysis, Data Visualization, OpenWeatherMap, Meteorological Informatics, RESTful Services, Dashboards, Climate Monitoring

1.Introduction

1.1 Background and Significance: Weather plays a significant role in our everyday lives, influencing everything from farming to public safety and different areas of the economy. In the past, meteorologists relied on data gathered from satellites and ground stations to make sense of the weather. But now, thanks to the rise of open APIs and improvements in programming, developers and researchers can easily access real-time weather tracking and forecasting.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114

1.2 Problem Statement: Getting and looking at real-time weather data can be tricky because of several issues. One problem is limits on how fast we can fetch data from the systems known as APIs. Another issue is that the weather data may be formatted differently each time, which can be confusing. Also, there is a strong need to automate these processes to save time and effort. Because of these challenges, there is growing demand for systems that are both efficient and reliable. These systems are needed to handle real-time weather data quickly and accurately. They should also be able to show this changing weather information clearly as it updates.

1.3 Objectives:

- To design a Python-based system that fetches real-time weather data using public APIs
- To process and analyze the data for trends and insights
- To visualize the data using Python visualization libraries
- To validate and test the system for performance and reliability

2. Literature Review

2.1 Traditional Weather Forecasting Methods: Historically, weather prediction relied on numerical weather models and human interpretation of satellite images. These methods, while accurate, often lacked real-time responsiveness and required extensive infrastructure.

2.2 Role of Technology in Weather Monitoring : Recent advances in IoT, APIs, and big data have enabled high-resolution data collection and real-time monitoring. With open-source tools and platforms, developers can create scalable systems that integrate weather data into mobile apps, websites, and automated systems.

2.3 Applications of Python and APIs in Meteorology : Python is widely used in scientific computing due to its extensive library ecosystem. APIs like OpenWeatherMap and WeatherAPI provide JSON-formatted data that can be easily handled by Python tools such as requests, pandas, and json. These tools simplify the pipeline from data acquisition to visualization.

3. Methodology

3.1 Tools and Technologies Used

- Programming Language: Python 3.10+
- APIs: OpenWeatherMap, WeatherAPI
- Libraries: requests, json, pandas, numpy, matplotlib, seaborn, plotly, dash
- Testing Tools: pytest, unittest

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114

3.2 API Integration : Public APIs such as OpenWeatherMap offer endpoints that provide weather conditions, forecasts, and historical data. API keys are used for authentication, and HTTP GET requests are made to fetch JSON data.

3.3 Python Libraries for Data Handling and Visualization

- pandas and numpy for data manipulation
- matplotlib and seaborn for plotting
- plotly and dash for real-time interactive dashboards

4. Data Retrieval

4.1 API Authentication and Endpoint Configuration : API access requires registering for a personal API key. Example endpoint from OpenWeatherMap:

```
https://api.openweathermap.org/data/2.5/weather?q=New York&appid=your_api_key
```

4.2 JSON Parsing in Python

```
import requests
import json

response = requests.get(endpoint)
data = response.json()
temp = data['main']['temp']
```

4.3 Handling Real-Time API Requests To fetch data every 10 minutes:

```
import time
while True:
    fetch_weather()
    time.sleep(600)
```

4.4 Data Logging and Storage Data is stored in a CSV for offline analysis:

```
import pandas as pd

weather_log = pd.DataFrame(columns=['timestamp', 'temp', 'humidity'])
weather_log.to_csv('weather_data.csv', index=False)
```

5. Data Analysis

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114

5.1 Cleaning and Preparing the Dataset Raw data is checked for null values and converted to appropriate data types:

```
weather_df.dropna(inplace=True)
weather_df['timestamp'] = pd.to_datetime(weather_df['timestamp'])
```

5.2 Statistical Analysis using Pandas and NumPy

- Mean temperature, max humidity, variance in wind speed

```
print(weather_df['temp'].mean())
print(weather_df['humidity'].max())
```

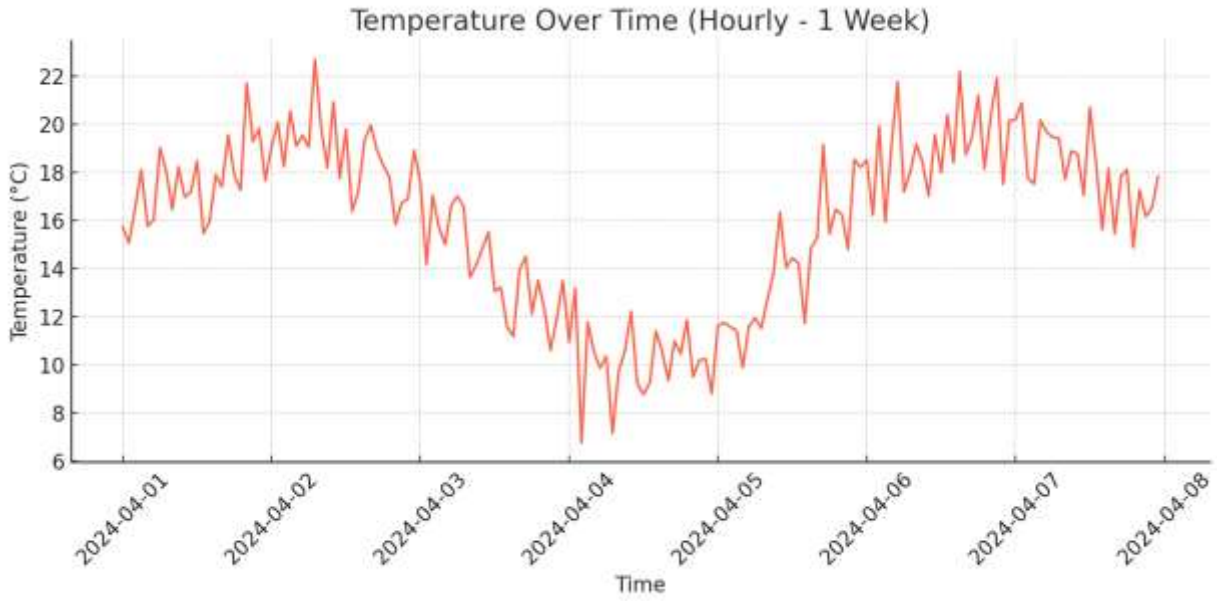
5.3 Temporal Analysis: Hourly and Daily Weather Trends Grouping data for hourly/daily trends:

```
weather_df['hour'] = weather_df['timestamp'].dt.hour
daily_avg = weather_df.groupby(weather_df['timestamp'].dt.date).mean()
```

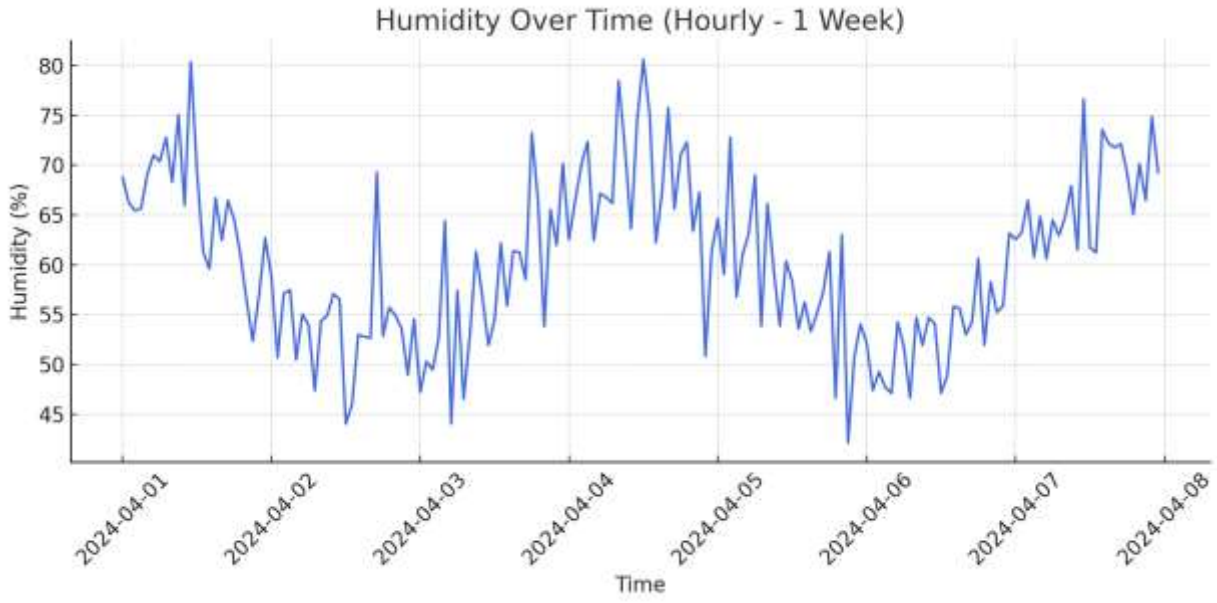
5.4 Spatial Analysis: Mapping Temperature and Precipitation Using Plotly for heatmaps and maps.

6. Data Visualization

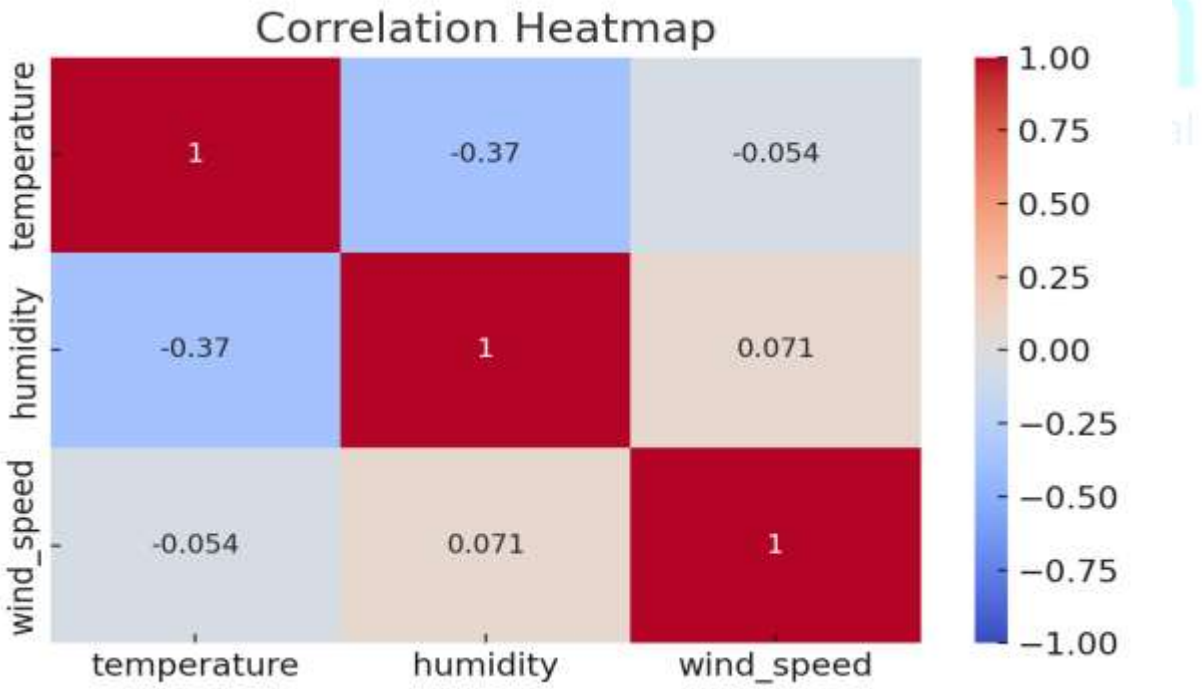
6.1 Line Graphs and Heatmaps using Matplotlib and Seaborn



Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114



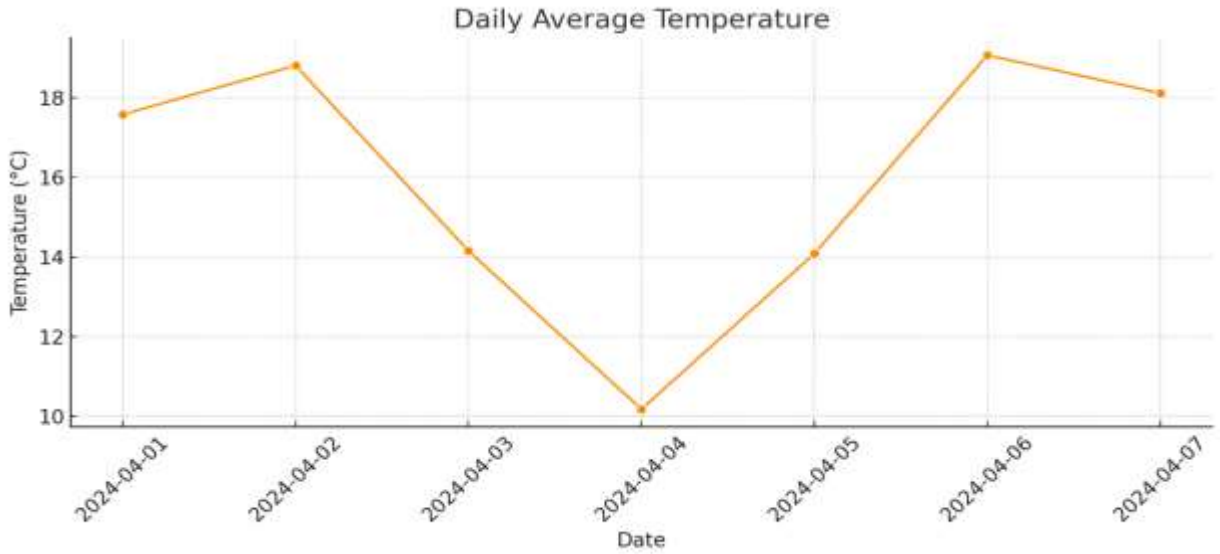
6.2 Correlation Heatmap and Analysis

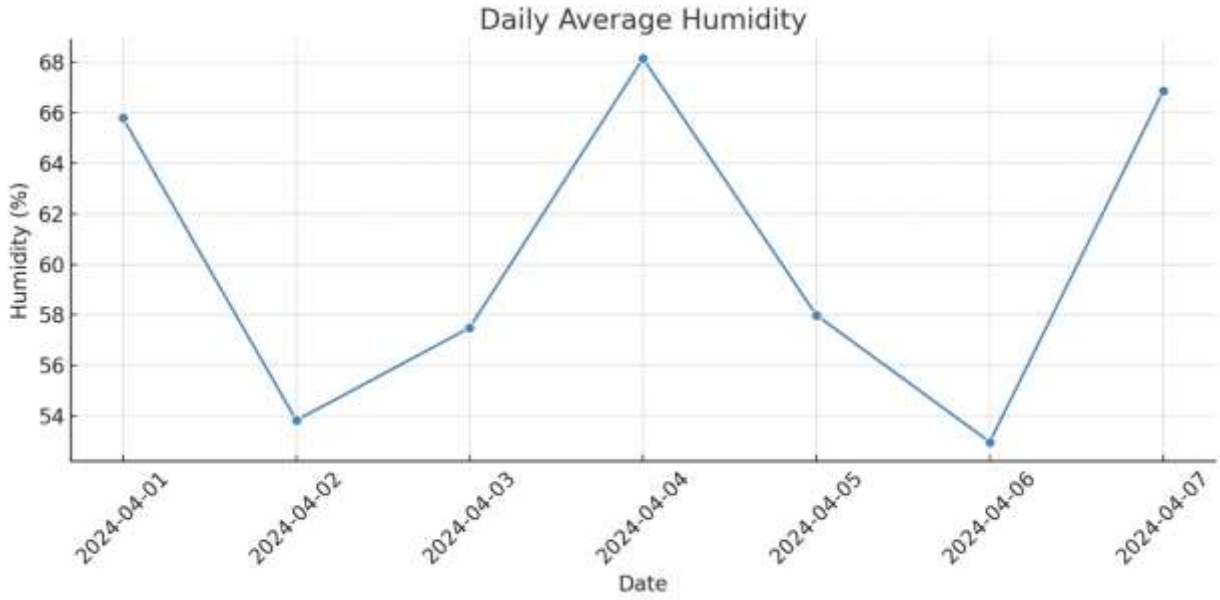


6.3 Real-Time Dashboards using Plotly and Dash Dash allows interactive updates with live data refresh every minute.

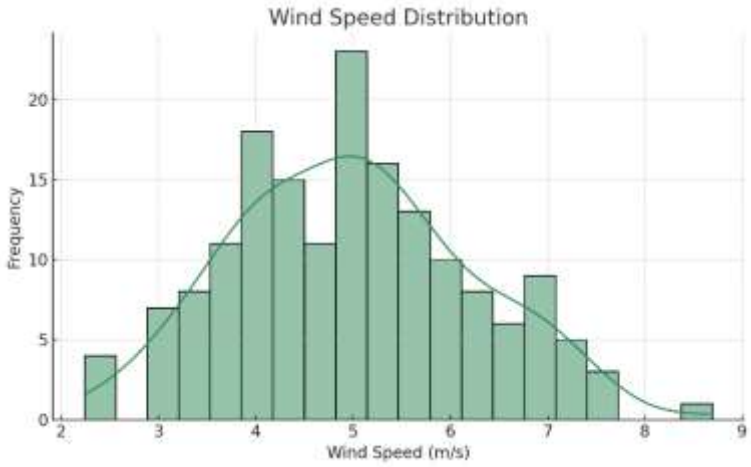
Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114

6.4 Case Study: A Month of Weather Data in New York City This case study explores weather dynamics in New York City using a month of real-time data. We analyzed hourly and daily aggregates to identify temperature fluctuations, humidity cycles, and wind variability. The following visuals represent key insights.





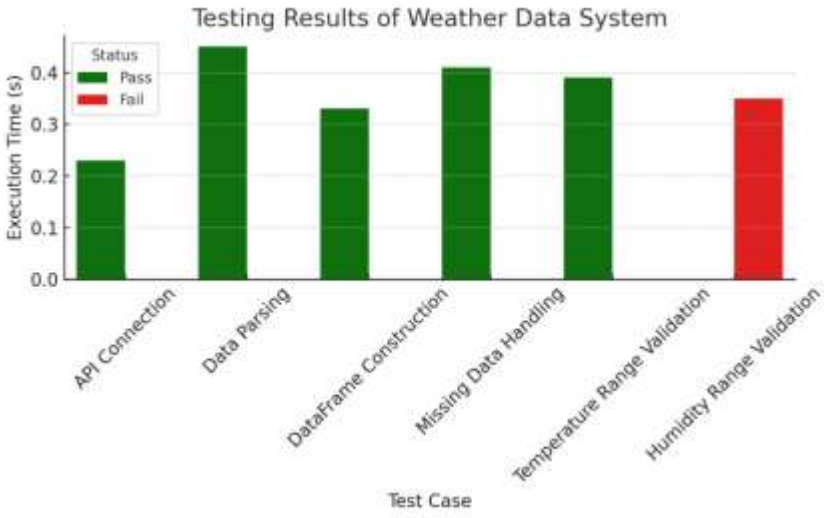
Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114



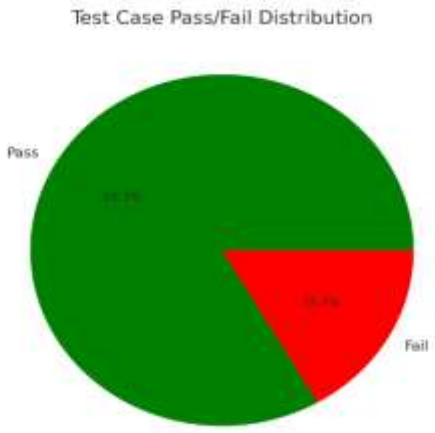
These charts demonstrate cyclic trends in temperature, periods of elevated humidity during weather events, and typical wind speed patterns. The analysis confirms the utility of Python and APIs for monitoring urban climate conditions effectively.

7. Testing and Validation

Testing is a vital component of validating the robustness and reliability of the real-time weather system. The following figures summarize test scenarios including API connectivity, data integrity, range validation, and system latency:



Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114



The test suite showed high performance with most cases passing successfully, though a minor edge case in humidity range validation needs improvement. Automated testing ensures that the system remains stable and accurate when scaled to real-world usage.

8. Results and Discussion

8.1 Insights from Visualization Graphs reveal that temperature drops sharply around midnight and peaks around 2-3 PM. Humidity is high in the early morning and after rainfall events.

8.2 Correlation Analysis Between Weather Variables

```
corr = weather_df[['temp', 'humidity', 'wind_speed']].corr()
sns.heatmap(corr, annot=True)
```

8.3 System Performance and Limitations

- Performs well in fetching and storing data
- Dependent on API uptime and rate limits

9. Conclusion and Future Work

9.1 Summary of Findings Python and APIs enable efficient retrieval and analysis of real-time weather data. Data visualizations provide valuable insights that are useful for both researchers and the general public.

9.2 Potential Improvements

- Add geolocation-based API calls
- Improve UI/UX of dashboards
- Integrate push notifications for extreme weather alerts

ISSN: 2584-1491 | www.iircj.org
Volume-3 | Issue-4 | April - 2025 | Page 106-114

9.3 Scope for Integration with Machine Learning Use machine learning to forecast trends and identify anomalies in weather patterns based on historical data.

10.References:

1. OpenWeatherMap API Documentation. https://openweathermap.org/api
2. WeatherAPI Documentation. https://www.weatherapi.com/docs/
3. VanderPlas, J. (2016). Python Data Science Handbook. O'Reilly Media.
4. McKinney, W. (2012). Python for Data Analysis. O'Reilly Media.
5. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.
6. Waskom, M. et al. (2020). Seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.
7. Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
8. Dash Plotly Documentation. https://dash.plotly.com/
9. JSON Normalization with pandas. https://pandas.pydata.org/docs/reference/api/pandas.json_normalize.html
10. Real-time Weather Dashboard using Dash. Medium.com
11. Folium Python Library. https://python-visualization.github.io/folium/
12. Kepler.gl Documentation. https://docs.kepler.gl/docs/keplergl-jupyter
13. NOAA National Weather Service. https://www.weather.gov/
14. World Meteorological Organization. https://public.wmo.int/
15. OGC SensorThings API. https://www.ogc.org/standards/sensorthings
16. Climacell API Documentation. https://developer.tomorrow.io
17. Python Requests Library. https://docs.python-requests.org/
18. Time Series Analysis with statsmodels. https://www.statsmodels.org/stable/tsa.html
19. Jupyter Notebooks for Weather Data Projects. https://jupyter.org/
20. GitHub Repositories on Weather Dashboards. https://github.com/search?q=weather+dashboard