Weather Data Report

Method

We used two primary sources for this Weather Data Report: OpenWeather API(https://openweathermap.org/api) and Tomorrow, io API (https://app.tomorrow.io/home). OpenWeather API was utilized for current weather data, providing temperature_max, temperature_min, humidity, and wind speed. Tomorrow.io was used for historical and forecast weather data, offering temperature_max, temperature_min, humidity, and wind speed. Regarding forecasting data, multiple rows showcased different forecast times within the same day, providing a detailed and granular view of expected weather conditions. Additionally, the data was integrated into a comprehensive dataset for comparative analysis. We utilized loop code to process the data efficiently, ensuring it was comprehensive and systematically organized.

To extract and represent the data information, we conducted feature engineering for all three datasets (current, historical, and forecast weather data). This involved parsing the timestamp data from the APIs to extract the day, month, and year. We also calculated the temperature range for each day by subtracting the minimum temperature from the maximum. These manipulations allowed us to perform more detailed analyses based on specific dates and temperature variations.

Challenges Faced

During the data collection process, we encountered several challenges: one significant issue was the occasional failure of the APIs to fetch data, which disrupted our data collection workflow. Additionally, there were limitations on the frequency of API requests, preventing us from refreshing the data as often as needed. The data from the APIs were not standardized to the same time zone, which added complexity to the data processing and analysis. The weather data from the different APIs were measured differently. One dataset was accessible to all users for no cost and included fundamental measurements, whereas the other offered more comprehensive measurements for a fee. Extra data cleaning and normalization were necessary to ensure that the datasets were consistent due to the absence of standardization.

The Pipeline

To ensure that our data pipelines for weather data run seamlessly daily, we have made sure each day, the pipelines will fetch and process the current_weather_data, weather data and forecast_weather_data, and the results will be ingested into our Microsoft SQL Server database. For example, the current_weather_data will be ingested into the "Current Weather Data" table within our database's "uploads" schema. The data will be inserted with precise data types for each column, including DECIMAL for geographic coordinates and temperature differences, VARCHAR for descriptive fields, and FLOAT for numerical weather parameters. The to_sql method from Pandas will ensure that the data is appropriately formatted and inserted with the

'replace' option to update the table daily. This process will allow us to maintain a current and accurate dataset in our SQL Server, facilitating ongoing analysis and reporting.

Conclusion

This report outlines a reliable methodology for collecting, processing and analyzing meteorological data using historical, current and forecast datasets. Applying feature engineering techniques transformed the data to facilitate detailed analysis. The project successfully implemented an integrated data pipeline despite challenges such as API outages, request frequency limitations, and time zone differences. The pipeline ensures that weather data is seamlessly fed into the Microsoft SQL Server database daily for ongoing analysis and accurate future weather forecasts. By addressing the challenges encountered and continuously improving the data pipeline, the company can increase the reliability and accuracy of its weather data services.