Welcome to

Python Flask: Web Development Basics to Machine Learning Deployment

By Md. Azizul Hakim
Lecturer in Daffodil International University

Weather App Using OpenWeather API Only

- The app acts as a **proxy** to fetch and display real-time weather data for a user-specified location.
- It relies entirely on the **OpenWeather API** to provide accurate, up-to-date weather information.
- No prediction or processing logic is implemented, as the app forwards API responses directly to the user after minor formatting.

Workflow

1. **User Input**: The user enters a city name (e.g., "London").
2. **API Request**: The backend sends a request to the OpenWeather API with the city name and API key.
3. **API Response**: The API returns weather details like temperature, humidity, wind speed, and weather conditions.
4. **Result Display**: The backend processes the JSON response and renders the information on a user-friendly interface.

Key Characteristics

- No **historical data** is used.
- Entirely dependent on the accuracy and availability of the API.
- Fast and straightforward to implement, with minimal computational overhead.

Weather Prediction App Using OpenWeather API and Machine Learning

- The app combines **real-time weather data from OpenWeather API** and **historical weather data** to train a machine learning model capable of predicting future weather conditions.
- The **ML model** is trained offline using historical data (e.g., temperature, humidity, wind speed, etc.) collected over time, potentially fetched from OpenWeather or other sources.
- The **prediction logic** happens within the app itself, which outputs future weather conditions, such as temperature trends, using the pre-trained ML model.

Workflow

1. **Training**:
   - The trained model is saved to disk (e.g., using `joblib` or `pickle`).
2. **User Input**:
   - The user inputs a city name to check both the current weather and a future prediction.
   - The current weather is fetched via the OpenWeather API.
3. **Prediction**:
   - The app uses the pre-trained model to predict future weather conditions (e.g., temperature for the next day).
   - Input features for prediction may include historical weather patterns for the city or extracted data from the real-time API response.
4. **Result Display**:
   - Both real-time weather data (from the API) and future predictions (from the ML model) are displayed in the app.

❖ **Enhanced Functionality**: Provides insights into future weather trends, not just the current state.

❖ **Reduced API Dependency**: Once the model is trained, the app relies less on the API, reducing costs and potential downtime from API limits.

Thank You