Weather Forecasting App

Project Description

If you are interested in visualizing weather data or the usage of OpenWeatherMap APIs, this project would help you achieve those goals. This project also encompasses the usage of Streamlit, a low code front end for Data Scientists.

Project Language(s)

Python

Prerequisite(s)

Python

Skills to be learned

PyOWM, Streamlit, Matplotlib, Data Visualization

Overview

Objective

You will be creating a single-page web application using Streamlit (a python framework) and matplotlib to display weather data visualizations.

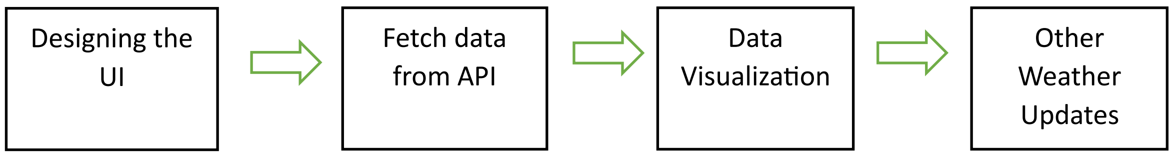
Project Context

We look at weather data and the future predicted the weather to plan our days accordingly. Having visualizations helps us understand that data better.

Developing this project using the **Streamlit** library we can create a responsive front-end which gives us more time to work on the actual back-end and the services we aim to provide.

This project is a good start for beginners in python and it gives a basic understanding of how to use APIs and related python frameworks.

Project Stages



High-Level Approach

• Setting up the Project environment

• Making the Streamlit front end

• Fetching the data from API calls

• Making the Data visualizations

• Adding other weather updates

• Deployment on Heroku

Task 1

Getting started with required files and dependencies

We are going to setup up the project's environment by setting up the application's starter files/folders. Also, the dependencies to be installed will be covered here.

By setting up the project environment we can get a low-level idea about how the project will look like and how the project stages would be broken down.

Requirements

In your working directory, you can create the following file structure (Kindly ignore LICENSE and README.md since these files might be required if you host the project files on GitHub).

• app.py will be our main file.

• requirements.txt is created so that other developers can install the correct versions of the required Python packages to run your Python code.

• The runtime.txt contains the version of python we need when deploying to Heroku.

• The Procfile is created for deployment using Heroku. It is not needed to run the app locally.

• setup.sh will contain commands which will be executed on the Heroku platform.

Install necessary libraries using pip package installer. It is recommended to use the following packages/libraries for this project.

streamlit==0.63.1

matplotlib==3.2.2

DateTime==4.3

pyowm==3.0.0

References

- DateTime
- Matplotlib
- Streamlit
- Pyowm

Tip

- While installing the packages (preferably while using the terminal) you can install multiple packages/libraries in a single line of command (by separating them by spaces) like this:

```
pip install lib1 lib2 lib3
```

Note

- **Streamlit** is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science.
- **PyOWM** is a client Python wrapper library for OpenWeatherMap web APIs. It allows quick and easy consumption of OWM data from Python applications via a simple object model and in a human-friendly fashion.
- **Matplotlib** is a plotting library for the Python programming language. It provides an object-oriented API for embedding plots into applications.
- **DateTime** provides a DateTime data type . DateTime objects represent instants in time and provide interfaces for controlling its representation without affecting the absolute value of the object.

Task 2

Creating basic Streamlit layout

Requirements

• Import relevant libraries as shown below:

```
import os

import pytz

import pyowm

import streamlit as st

from matplotlib import dates

from datetime import datetime

from matplotlib import pyplot as plt
```

• Get the api key from the OpenWeatherMap website and use it as follows:

```
owm=pyowm.OWM('your-api-key')

mgr=owm.weather_manager()
```

• For the streamlit frontend we will need a title and a placeholder text:

```
st.title("Weather Forecast")

st.write("### Write the name of a City and select the Temperature Unit from the sidebar")
```

• Now we will input the city name and using store it in a variable called the place

```
place=st.text_input("NAME OF THE CITY :", "")

if place == None:

st.write("Input a CITY!")
```

• Now we will create two selection forms as follows :

```
unit=st.selectbox("Select Temperature Unit",("Celsius","Fahrenheit"))
```

• To run the code use the following command :

```
streamlit run <filename>
```

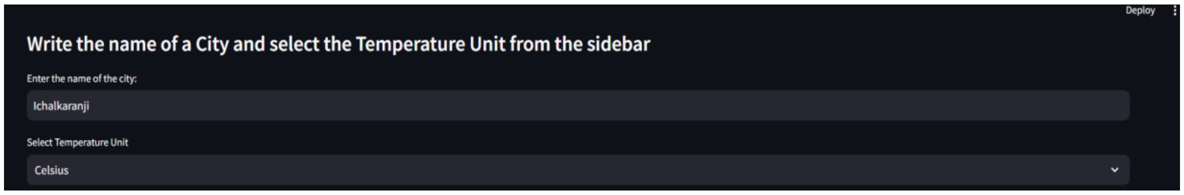
References

• Streamlit Documentation

• OpenWeatherMap API

Expected Outcome

Finally, our web app has the front-end ready. It will look like this :


Deploy

Write the name of a City and select the Temperature Unit from the sidebar

Enter the name of the city:

Ichalkaranji

Select Temperature Unit

Celsius

Task 3

Fetching the API data

Now that our front-end is done we need to fetch the data using the PyOWM API so that we

can visualize it.

Requirements

- The OpenWeatherMap free tier gives you access to 5-day forecasts. The forecasts contain the weather data in three-hour intervals.
- The methods for retrieving the forecast are:
  - forecast_at_place('Los Angeles, US', '3h')
  - forecast_at_id(5391959, '3h')
  - forecast_at_coords(lat=37.774929, lon=-122.419418, interval='3h')
- Create an OWM object.
- Next, we get a WeatherManager instance.
- Retrieve a 5-day forecast data with intervals of 3 hours.

References

• PyOWM Documentation

Note

• At this point you might be wondering what's the difference is between the forecaster

object and the forecast object. A *Forecaster* object is returned by the

forecast_at_place() method, containing a *Forecast*: this instance encapsulates

Weather objects corresponding to the provided granularity.

• The Forecast class gives you access to the actual values in the forecast, such as the

temperature, amount of rainfall, etc. while the Forecaster class provides a high-level

interface to the forecast so you can check whether or not there are specific weather

conditions (such as rain or clouds) in the forecast. We'll explore some of the methods for the Forecaster class in later steps.

Expected Outcome

You are expected to use the PyOWM API to fetch the weather forecast data.

Task 4

Plot the Temperature

Now that we have a function to retrieve the weather, we're ready for the fun part, plotting

the temperature!

Requirements

- Create functions for line chart.
- Pass the days , minimum temperature and maximum temperature as parameters.
- Use the plt.plot(days,temp_min) and plt.plot(days,temp_max) for plotting minimum and maximum

temperatures on the bar graph respectively.

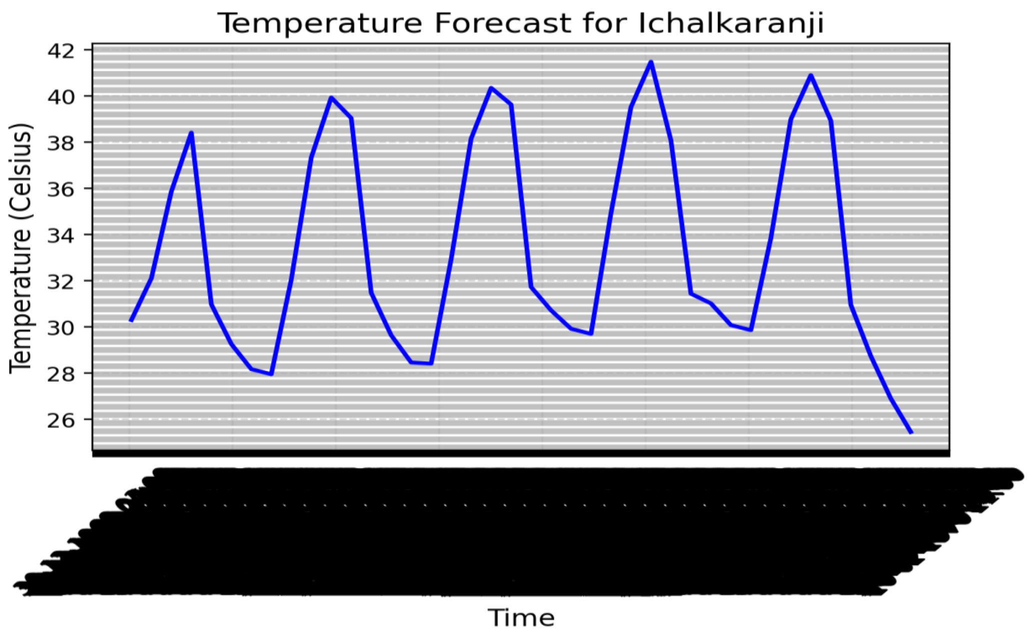
- At this point the plot should showing the forecast for the next five days.

Bring it On!

Optionally you can add the temperatures as labels on the line graphs.

Expected Outcome

The line graphs should look like this :



Task 5

Adding the additional weather updates

Now that we have a function to plot the weather data, we can now include details like the impending weather changes, cloud coverage, wind speed, and sunrise and sunset times.

Requirements

• Python Open Weather Map provides methods to check for certain weather conditions in a forecast.

• For example, the following snippet checks if there is rain in this week's forecast for LA:

```
forecaster = mgr.forecast_at_place('Los Angeles, US', '3h')

print(forecaster.will_have_rain())
```

• Knowing what time the sun rises and sets is great information to have when planning your day! Let's see how we can get these times using PyOWM.

sunrise_time(): Returns the GMT time of sunrise

sunset_time(): Returns the GMT time of sunset

For example :

```
print(weather.sunrise_time())
```

- PyOWM gives you access to other weather information in addition to the temperature.

Some of the information you can get about is:

- wind
- clouds
- humidity

The values in the clouds and humidity properties are of type `int`. The values represents the percentage of cloud cover and the humidity percentage respectively.

For example, to print the humidity after obtaining a `Weather` object, you could write:

```
humidity = weather.humidity

print(f'The current humidity is {humidity}%')
```

- All we have to do is to integrate all these functions into the main file.

Note

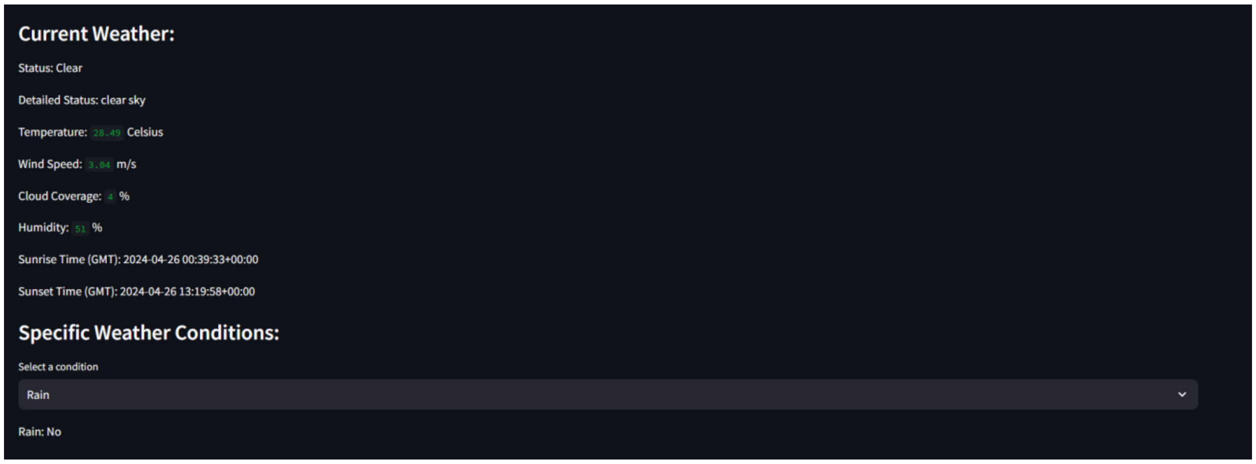
• You'll notice that sunrise time looks a little cryptic. That's because the value returned is in UNIX time. PyOWM allows you to add a parameter to specify ISO time as well, which is more easily readable. For example, check out the snippet below:

print(weather.sunrise_time(timeformat='iso'))

• Keep in mind that the times are for the GMT time zone, so they may still look a little strange. The solution is presented by the pytz module. pytz is a Python module that simplifies working with time zones.

Expected Outcome

You should be able to access the following types of data and represent them as given.



Current Weather:

Status: Clear

Detailed Status: clear sky

Temperature: `28.49` Celsius

Wind Speed: `3.04` m/s

Cloud Coverage: `4` %

Humidity: `51` %

Sunrise Time (GMT): 2024-04-26 00:39:33+00:00

Sunset Time (GMT): 2024-04-26 13:19:58+00:00

Specific Weather Conditions:

Select a condition

Rain

Rain: No

Task 6

Deploying the project on Heroku

Now that your project is complete, feel free to deploy it on Heroku for free!

Requirements

• Create a Heroku account.

• Run pip freeze > requirements.txt in the terminal. This creates a text file containing all our dependencies.

• Create a Procfile in the working directory. This file will contain commands to be run on the Heroku cloud.

• Run sudo snap install --classic heroku on terminal to install the Heroku CLI.

• In the Heroku terminal, run the commands below:

```
heroku create unique-name-here

git add .

git commit -m 'Initial app template'

git push heroku master
```

References

• Heroku CLI

Note

• The URL should be generated after following the aforementioned steps. It should look like https://<unique-name-here>.herokuapp.com.

• Heroku has some limitations on free plans. However, you can upgrade by paying a fee.

• In case of deployment errors, Heroku maintains a log of it for you to check out.

• You may use any other hosting services too if you are uncomfortable with using Heroku.

Expected Outcome

Finally, our web app is deployed and can be accessed by anyone in the world.