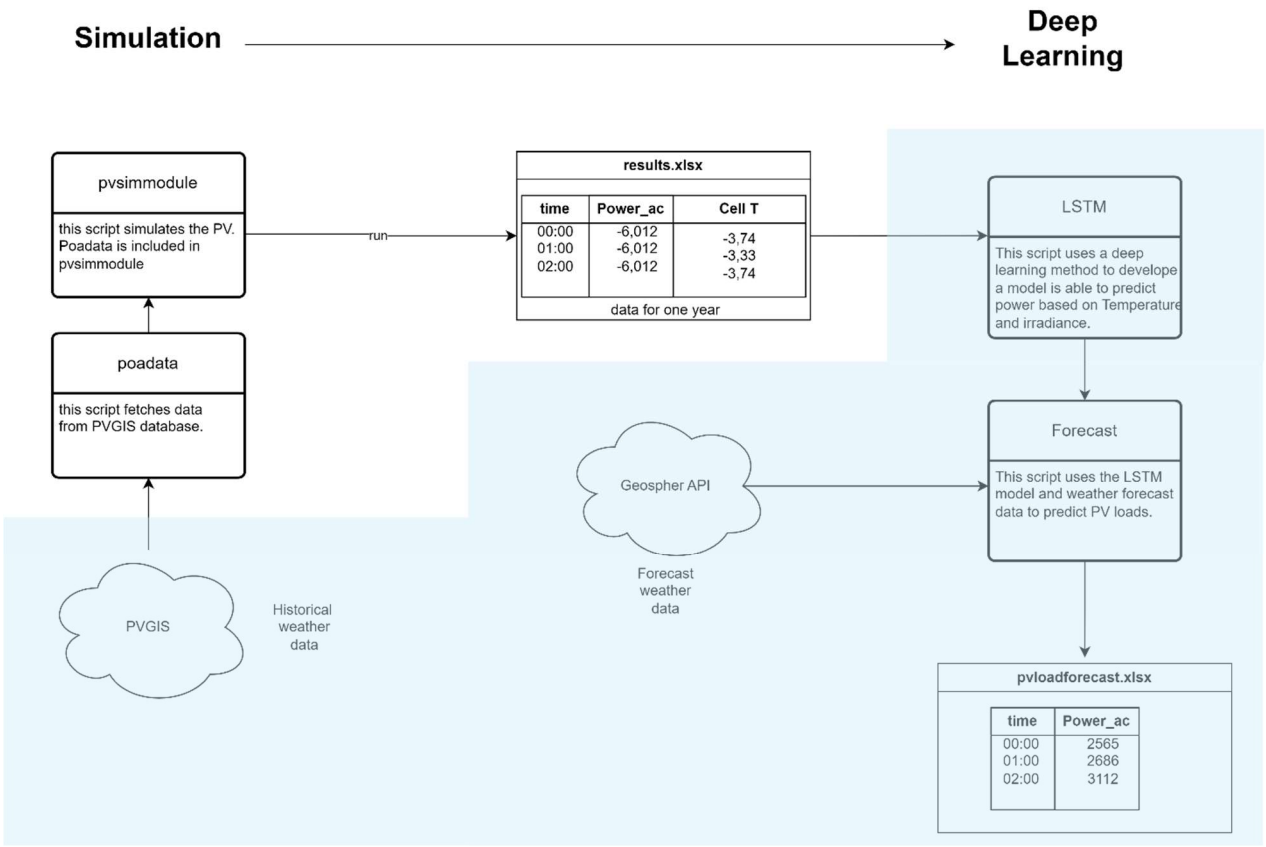
PV Simulation and forecast based on deep learning model (LSTM)

A project by Michael Grün



In addition to addressing the **initial subject** as requested, **further advancements** were made. These enhancements, **highlighted in blue**, extend beyond the original scope and provide additional insights and value to the research.

Geosphere API implementation with python

This script is designed to help us fetch **weather forecast** data from an online source and process it into a readable **Excel** file. Here's a step-by-step explanation of how it works:

1. **Getting User Input**:
    - The script first determines the time range for the weather forecast. It asks for how many hours into the future we need the forecast and calculates the start and end times.
2. **Fetching Weather Data**:

    - Using the specified location (latitude and longitude for Leoben EVT), the script sends a request to a weather data API (an online service that provides weather information).
    - The API responds with detailed weather forecast data, which the script saves as a JSON file on our computer.
3. **Processing the Data**:
    - The script reads the saved JSON file and processes the data. It extracts important information such as temperature, wind speed, sunshine duration, and global radiation.
    - This data is organized into a structured format (a DataFrame) that is easy to work with and analyze.
4. **Creating an Excel File**:
    - The processed data is then saved into an Excel file. This makes it easy for us to view, share, and use the forecast data in our reports and presentations.
5. **Running the Script**:
    - The script is run automatically, fetching the latest weather data and updating the Excel file without needing manual intervention each time.

Benefits:

- **Efficiency**: Automates the collection and processing of weather data, saving time.
- **Accuracy**: Ensures we have the latest and most accurate weather forecasts.
- **Accessibility**: The final output is in Excel format, which is easy to use and understand.

By using this script, we streamline our process of gathering and analyzing weather forecast data, which supports better decision-making in our projects.

```
# -*- coding: utf-8 -*-
"""
Author: Michael Grün
Email: michaelgruen@hotmail.com
Description: This script fetches weather forecast data from an API and
processes it to create an Excel file
             with the forecast data for a specified number of hours.
Version: 1.0
Date: 2024-05-04
"""

import json
import pandas as pd
import requests
from datetime import datetime, timedelta

# Latitude and lonfitude for Leoben EVT
lat_lon = "47.38770748541585,15.094127778561258"

def get_user_input(hours: int) -> tuple:
    """
```

```
    Get user input for forecasting start and end times.

    Args:
        hours (int): The number of hours to forecast.

    Returns:
        tuple: A tuple containing the forecasting start time and end time.
    """
    current_time = datetime.now().replace(minute=0, second=0,
microsecond=0)

    fc_start_time = current_time.strftime("%Y-%m-%dT%H:%M")
    fc_end_time = (current_time + timedelta(hours=hours)).strftime("%Y-%m-
%dT%H:%M")

    return fc_start_time, fc_end_time


def fetch_weather_data(lat_lon, fc_start_time, fc_end_time):
    """
    Fetch weather forecast data from the API.

    Args:
        lat_lon (str): Latitude and longitude for the location.
        fc_start_time (str): Forecasting start time in the format "YYYY-MM-
DDThh:mm".
        fc_end_time (str): Forecasting end time in the format "YYYY-MM-
DDThh:mm".

    Returns:
        dict: The fetched weather forecast data as a dictionary.
    """
    url =
f"https://dataset.api.hub.geosphere.at/v1/timeseries/forecast/nwp-v1-1h-
2500m?lat_lon={lat_lon}&parameters=grad&parameters=t2m&parameters=sundur_ac
c&parameters=v10m&start={fc_start_time}&end={fc_end_time}"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        file_path = "weather_forecast.json"
        with open(file_path, "w") as json_file:
            json_file.write(response.text)
        print("JSON data saved successfully.")
        return data
    else:
        print(f"Failed to fetch data. Status code: {response.status_code}")
        return None


def process_weather_data(data):
    """
    Process the fetched weather forecast data and create a DataFrame.

    Args:
        data (dict): The fetched weather forecast data.

    Returns:
        pandas.DataFrame: A DataFrame containing the processed weather
data.
    """
    with open('weather_forecast.json') as f:
        data = json.load(f)
```

```
    df = pd.json_normalize(data['features'], meta=['properties',
['parameters']])
    time_list = data['timestamps']

    df1 = df[['properties.parameters.grad.data',
              'properties.parameters.t2m.data',
              'properties.parameters.sundur_acc.data',
              'properties.parameters.v10m.data']]

    list1 = list(df1['properties.parameters.grad.data'][0])
    list2 = list(df1['properties.parameters.t2m.data'][0])
    list3 = list(df1['properties.parameters.sundur_acc.data'][0])
    list4 = list(df1['properties.parameters.v10m.data'][0])

    df2 = pd.DataFrame()
    df2['timestamp'] = time_list
    df2['surface global radiation [J/m²]'] = list1
    df2['Temperture 2m above ground [°C]'] = list2
    df2['sunshine duration accumulated [s]'] = list3
    df2['wind speed northern direction [m/s]'] = list4

    return df2


def save_data_to_excel(df):
    """
    Save the processed weather data to an Excel file.

    Args:
        df (pandas.DataFrame): The DataFrame containing the processed
weather data.
    """
    with pd.ExcelWriter("forecast_data.xlsx") as writer:
        df.to_excel(writer, sheet_name='Forecast_Data1')
    print("Data saved to Excel file successfully.")


def main():
    """
    Main function to run the script.
    """


    fc_start_time, fc_end_time = get_user_input(24)

    weather_data = fetch_weather_data(lat_lon, fc_start_time, fc_end_time)
    if weather_data is not None:
        df = process_weather_data(weather_data)
        print(df)
        save_data_to_excel(df)


if __name__ == "__main__":
    main()
```

Snipped of Excel Export

	timestamp	surface global radiation [J/m²]	Temperture 2m above ground [°C]	sunshine duration accumulated [s]	wind speed northern direction [m/s]
0	2024-05-04T16:00+00:00	16736851,2	15,7	12640,6	1,9
1	2024-05-04T17:00+00:00	16874672	14,5	12640,6	-0,3
2	2024-05-04T18:00+00:00	17096265,6	13,2	12640,6	-1,2
3	2024-05-04T19:00+00:00	17100320	13	12640,6	-1,3
4	2024-05-04T20:00+00:00	17100320	12,5	12640,6	-0,6
5	2024-05-04T21:00+00:00	17100320	9,6	12640,6	-0,1
6	2024-05-04T22:00+00:00	17100320	9,6	12640,6	-0,5
7	2024-05-04T23:00+00:00	17100320	8	12640,6	-0,5
8	2024-05-05T00:00+00:00	17100320	6,7	12640,6	-0,9
9	2024-05-05T01:00+00:00	17100320	5,7	12640,6	-0,9
10	2024-05-05T02:00+00:00	17100320	5,1	12640,6	-0,6
11	2024-05-05T03:00+00:00	17100320	4,5	12640,6	-0,6
12	2024-05-05T04:00+00:00	17112480	4,3	12640,6	-0,5
13	2024-05-05T05:00+00:00	17486755,2	7,1	16000,3	-0,4
14	2024-05-05T06:00+00:00	18512302,4	10,9	19240,9	0,2
15	2024-05-05T07:00+00:00	20193168	13	22495,2	-0,2
16	2024-05-05T08:00+00:00	22488822,4	14,9	25827,6	0,3
17	2024-05-05T09:00+00:00	25319544	17,5	29169,7	0,3
18	2024-05-05T10:00+00:00	28432659,2	19,8	32174,2	1,7
19	2024-05-05T11:00+00:00	31620092,8	20,9	34661,8	1,6
20	2024-05-05T12:00+00:00	34758880	21,3	36093,8	1,7
21	2024-05-05T13:00+00:00	37612572,8	21,4	37648,8	2,5
22	2024-05-05T14:00+00:00	39740678,4	21,1	37648,8	1
23	2024-05-05T15:00+00:00	41067539,2	19,8	37648,8	-0,7
24	2024-05-05T16:00+00:00	41759344	16,6	38109,2	0,5

Figure 1: Here you can see a representative export of the excel file

- **Timestamps**: Each record is associated with an hourly timestamp, detailing when the measurements were taken, ranging from May 4, 2024, at 16:00 to May 5, 2024, at 16:00.
- **Surface Global Radiation [J/m^2]**: This column measures the total amount of solar energy hitting a square meter of the earth's surface, recorded in joules per square meter. The values fluctuate across different hours, generally increasing during daylight hours.
- **Temperature 2m Above Ground [°C]**: This column records the air temperature measured two meters above the ground in degrees Celsius. It shows a natural variation, typically cooler during early morning hours and warmer in the midday.
- **Sunshine Duration Accumulated [s]**: This represents the total accumulated duration of sunshine in seconds. The data suggests a constant value over certain periods, likely indicating total sunshine up to that hour.
- **Wind Speed Northern Direction [m/s]**: This column lists wind speed values in meters per second, indicating the intensity of wind blowing from the north. The data includes both positive and negative values, where negative might indicate wind blowing from the south or a data entry error.

If you have any questions or need further clarification, please feel free to contact me at michaelgruen@hotmail.com