Project work, part 1 - Dashboard basics

Orie Kimura

~~October 3, 2025~~

October 16, 2025

General

1. Brief description of AI usage
   I used AI throughout the entire project. My work was divided into three main parts: setting up the tools, the actual coding, and finally, writing the report.
   When I was installing and preparing my working environment, I struggled a lot to understand the relationships between Miniconda, Jupyter Notebook, Streamlit, VS Code, and GitHub. It was difficult to figure out their specific functions and how they should be connected in my project. AI was a great help at this stage.
   Once I got to the coding phase, I used AI for several things:
   Explanations: If I came across a concept I didn't understand, I'd ask the AI to explain it in a simpler way.
   Debugging: AI helped me to fix errors. I'd give it the error message, and it would explain what the error meant, why it happened, and give me suggestions on how to fix it.
   New Solutions: The AI also helped me find alternative ways to solve a problem, which taught me to think more creatively.
   And finally, when I was writing the report, I used AI to double-check my language and make sure the text was clear and easy to read.

2. Log describing the compulsory work
   a. Installing and Preparing My Working Environment
   MiniConda
   The latest version of Miniconda 3 is installed on my PC. After creating my Conda environment for the project work in the Anaconda Powershell prompt, I installed the following:
   Python : conda install python==3.12.10 pip
   Jupyter : conda install jupyter
   Streamlit : conda install -c conda-forge streamlit
   VS Code
   The latest version of VS Code is installed with the Python and Jupyter extensions. It's important to open the folder where the .py files are saved.
   Accounts and Tools

GitHub: I created an account on https://github.com/. The latest version of GitHub Desktop is installed on my PC.
Jupyter Notebook: I created an account on https://jupyter.org/.
Streamlit: I created an account on https://share.streamlit.io/.

Useful Commands

To run Jupyter notebook: *(my_env) PS C:\Orie\IND320> jupyter notebook (Jupyter notebook coming up on http://localhost:8888/tree/)*
*To start Streamlit and run a .py file on local host: *(my_env) PS C:\Orie\IND320> streamlit run ..py** *(Streamlit coming up on http://localhost:8501/)*

b. Coding

After installing and testing all my tools, I started working on the project tasks.

Jupyter Notebook

The tasks were copied and pasted in Markdown into the notebook.
Then, I wrote and tested the code for each task in Jupyter.
Streamlit I adapted the examples from the lecture to complete the tasks and ran them on my PC.
Next, I pulled the files to my GitHub repository, and then deployed them to Streamlit.
Both the GitHub repository and the Streamlit app were changed to public.
Streamlit Cloud Deployment The deployment process to Streamlit Cloud followed these steps:
Log in to Streamlit Cloud: Access the Streamlit Community Cloud website and log in with my GitHub account.
Create a New App: Select the option to create a new app.
Connect to GitHub: Choose the GitHub repository where my Streamlit app code resides.
Make Changes in VS Code: Modify the Streamlit app code in VS Code.
Commit and Push to GitHub: Commit the changes and push them to my GitHub repository.
Streamlit Cloud Updates: Streamlit Cloud automatically detects changes in my linked GitHub repository and redeploys my application, updating the live version.

c. Submitting the Assignment

I attempted to export my Jupyter notebook file to a PDF file using 'nbconvert[webpdf]', including page numbers and ensuring all Python code was exported without turning/wrapping lines, but I couldn't figure out the correct configuration. Therefore, I exported the file as an HTML file and then converted it to a PDF format.

3. links to your public GitHub repository and Streamlitapp (see below) for the compulsory work.
   Originally, I submitted the assignments at the following links:
   https://github.com/oriekimura123/IND320-Projectwork-part1
   https://oriekimura-ind320-projectwork-part1.streamlit.app/
   However, I realized that reorganization of my working folders is necessary, and I am providing updated files at new links.

https://github.com/oriekimura123/IND320-Projectwork
https://oriekimura-ind320-projectwork.streamlit.app/

Tasks, Jupyter Notebook

1. Read the supplied CSV file (open-meteo-subset.csv) using Pandas.

In [1]:
```
import sys
# Add the repository root to the path so Python can find 'core'
sys.path.append('..')

# Import the specific function for reading CSVs with Pandas
from core.data_loader import load_csv_with_pandas

# Read the supplied CSV file (open-meteo-subset.csv) using Pandas.
import pandas as pd
meteo_subset_DF = load_csv_with_pandas('open-meteo-subset.csv')
```

2. Print its contents in a relevant way.

In [2]:
```
meteo_subset_DF.head()
```

Out[2]:

	time	temperature_2m (°C)	precipitation (mm)	wind_speed_10m (m/s)	wind_gusts_10m (m/s)	wind_directi
0	2020-01-01 00:00:00	-2.2	0.1	9.6	21.3	
1	2020-01-01 01:00:00	-2.2	0.0	10.6	23.0	
2	2020-01-01 02:00:00	-2.3	0.0	11.0	23.5	
3	2020-01-01 03:00:00	-2.3	0.0	10.6	23.3	
4	2020-01-01 04:00:00	-2.7	0.0	10.6	22.8	

In [3]:
```
meteo_subset_DF.info()
```

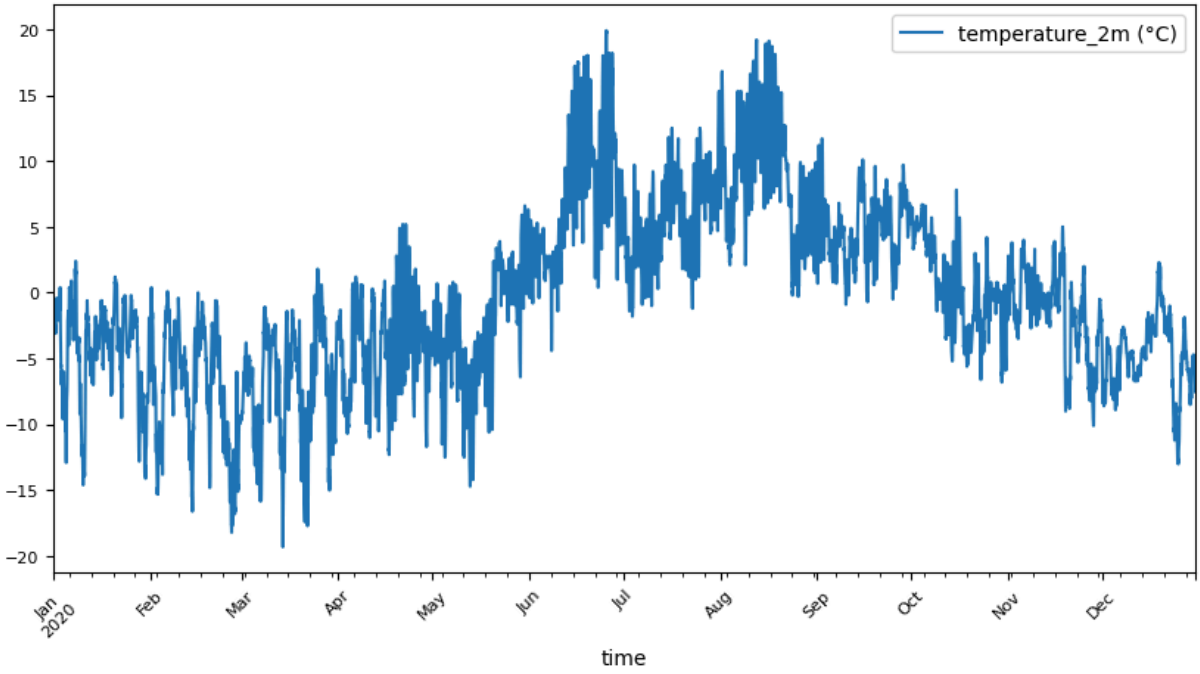
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   time                    8760 non-null   datetime64[ns]
 1   temperature_2m (°C)     8760 non-null   float64       
 2   precipitation (mm)      8760 non-null   float64       
 3   wind_speed_10m (m/s)    8760 non-null   float64       
 4   wind_gusts_10m (m/s)    8760 non-null   float64       
 5   wind_direction_10m (°)  8760 non-null   int64         
dtypes: datetime64[ns](1), float64(4), int64(1)
memory usage: 410.8 KB
```

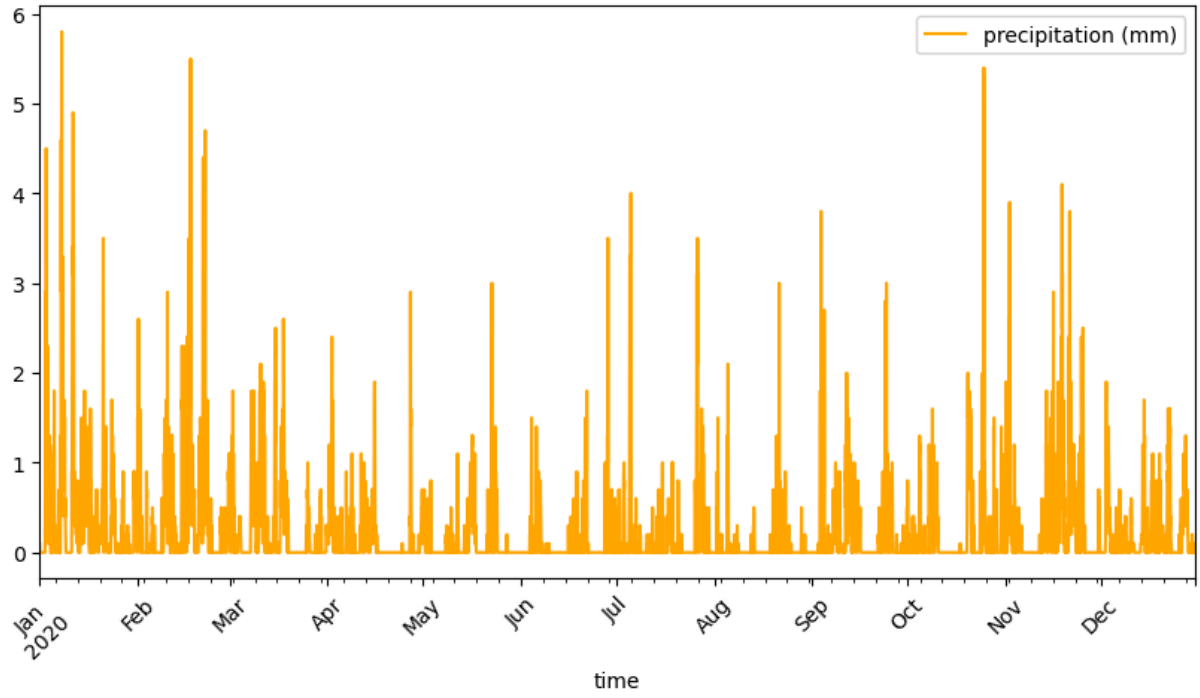
In [4]:
```
meteo_subset_DF.columns
```

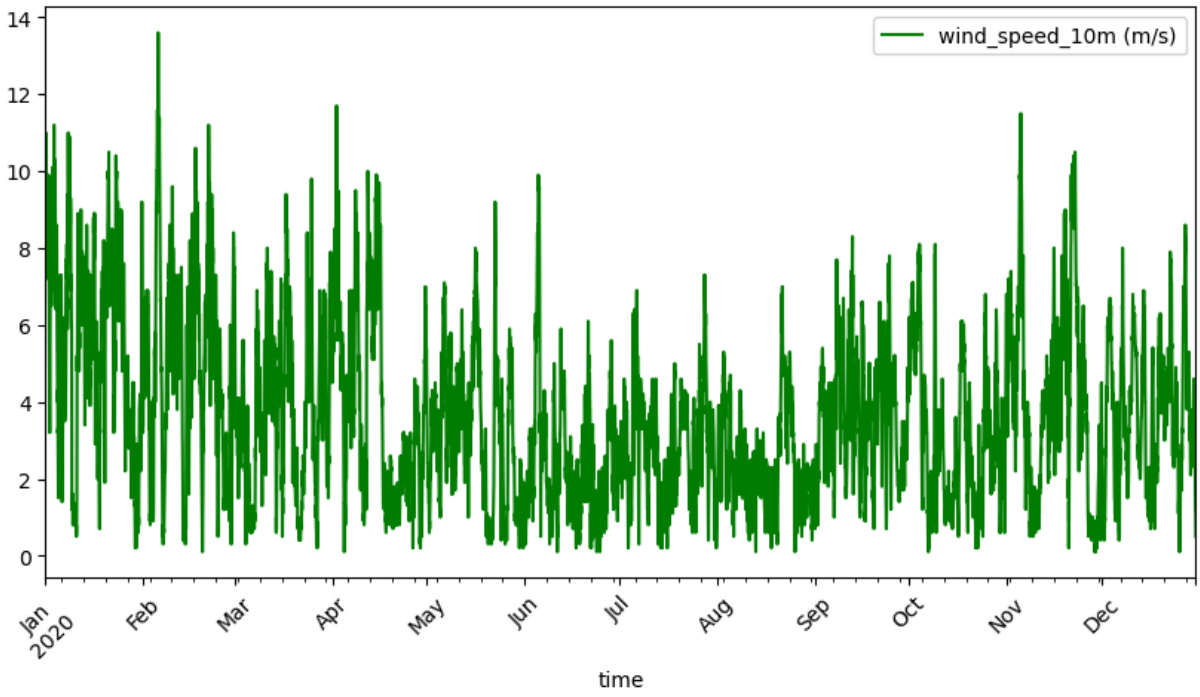
Out[4]:
```
Index(['time', 'temperature_2m (°C)', 'precipitation (mm)',
       'wind_speed_10m (m/s)', 'wind_gusts_10m (m/s)',
       'wind_direction_10m (°)'],
      dtype='object')
```

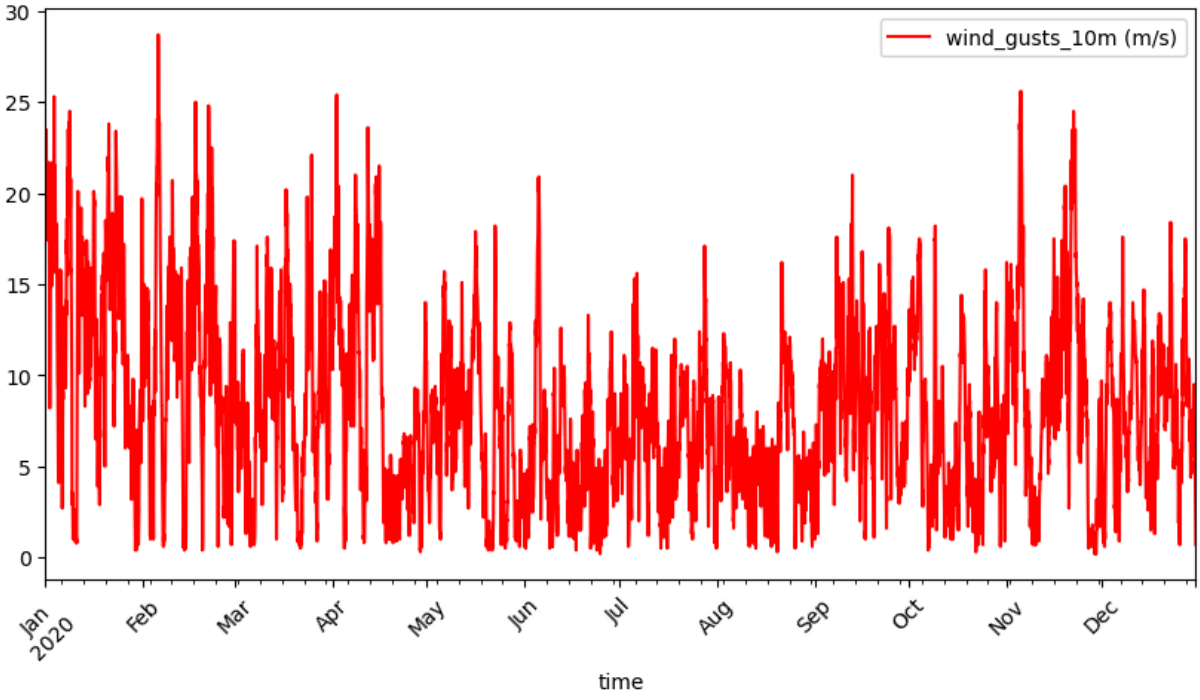
3. Plot each column separately.

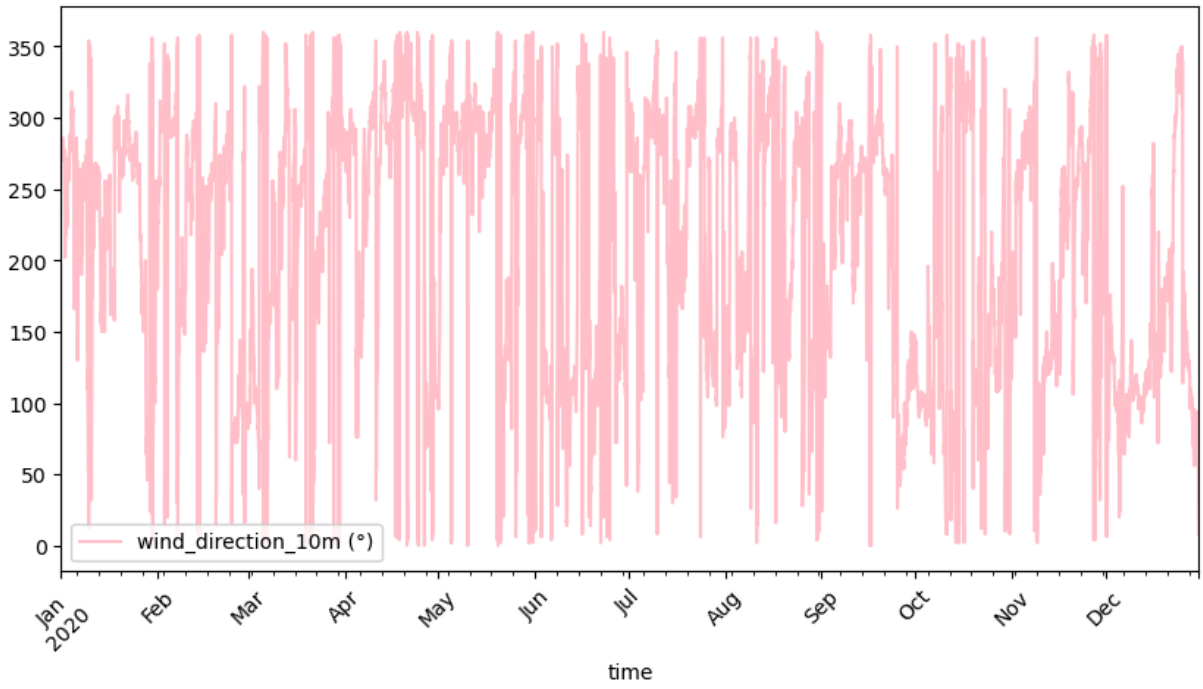
In [ ]:
```
# pd.options.plotting.backend = "matplotlib"
import matplotlib.pyplot as plt

# All tables have a custom size (10 inches wide by 6 inches tall),
# x-labels are rotated by 45 degrees and fontsize = 8
# Each observation has a different color

# Show temperature_2m (°C)
plt.show(meteo_subset_DF.plot(x='time', y='temperature_2m (°C)',
                              rot=45, figsize=(10, 5), fontsize=8))

# Show precipitation (mm)
plt.show(meteo_subset_DF.plot(x='time', y='precipitation (mm)', color='orange',
                              rot=45, figsize=(10, 5)))

# Show wind_speed_10m (m/s)
plt.show(meteo_subset_DF.plot(x='time', y='wind_speed_10m (m/s)', color='green',
                              rot=45, figsize=(10, 5)))

# Show wind_gusts_10m (m/s)
plt.show(meteo_subset_DF.plot(x='time', y='wind_gusts_10m (m/s)', color='red',
                              rot=45, figsize=(10, 5)))

# Show wind_direction_10m (°)
plt.show(meteo_subset_DF.plot(x='time', y='wind_direction_10m (°)', color='pink',
                              rot=45, figsize=(10, 5)))
```


temperature_2m (°C)
20
15
10
5
0
−5
−10
−15
−20
Jan
2020
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
time


precipitation (mm)
6
5
4
3
2
1
0
Jan
2020
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
time


wind_speed_10m (m/s)
14
12
10
8
6
4
2
0
Jan
2020
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
time


wind_gusts_10m (m/s)
30
25
20
15
10
5
0
Jan
2020
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
time

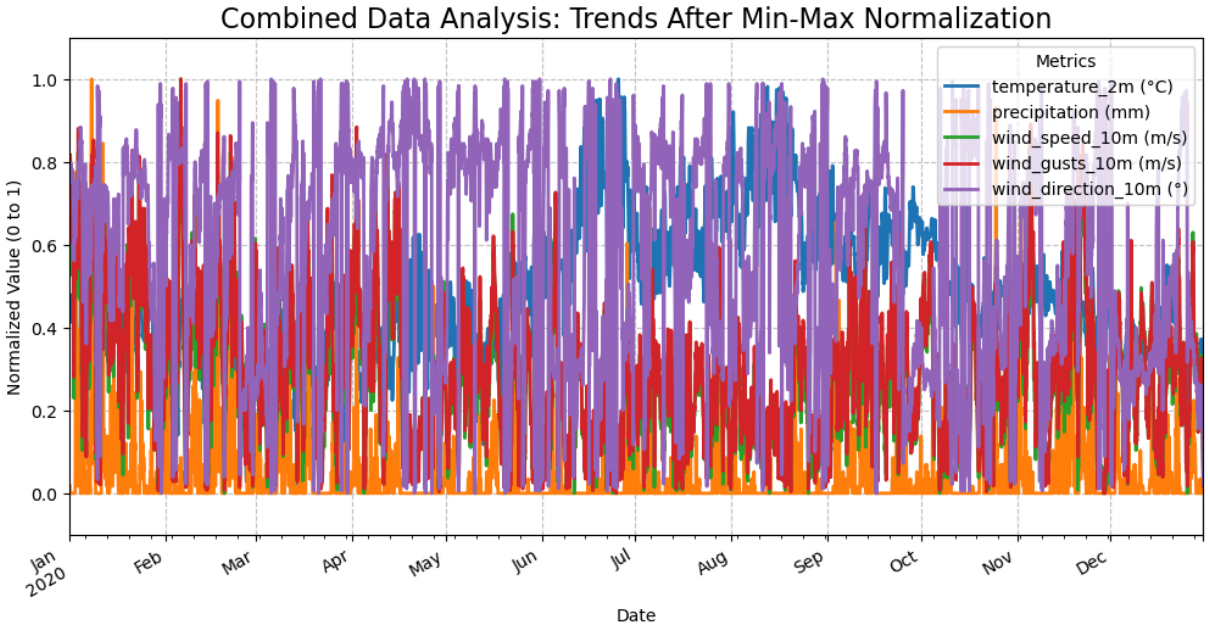


4. Plot all columns together. Consider how to make this natural, given that the scales are different.

In [ ]:
```python
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd # Assuming you have this imported

#---new ---plotting together of the columns
# (expected one overlapping plot) using x-y-normalization
print("\n--- Plotting All Columns Together Using Min-Max Normalization ---")

# Columns to normalize
# Get a list of the columns to plot, excluding the 'time' column.
# and add "All Columns" option
chart_columns = [col for col in meteo_subset_DF if col != 'time']
options_columns = ["All Columns"] + chart_columns

# Create a new DataFrame for normalized data
meteo_normalized_DF = meteo_subset_DF.copy()

for col in chart_columns:
    min_val = meteo_normalized_DF[col].min()
    max_val = meteo_normalized_DF[col].max()

    # Handle the case where max and min are the same
    if max_val == min_val:
        # Overwrite the original column data with 0
        meteo_normalized_DF[col] = 0
    else:
        # Apply Min-Max Scaling and overwrite the original column
        meteo_normalized_DF[col] =
        (meteo_normalized_DF[col] - min_val)/ (max_val - min_val)
```

```
# Plot the normalized data
fig_normalized, ax = plt.subplots(figsize=(12, 6))

# --- New: Define the desired date format (e.g., 'Jan 2024') ---
df_for_plotting = meteo_normalized_DF.set_index('time')
month_year_formatter = mdates.DateFormatter('%b %Y')
ax.xaxis.set_major_formatter(month_year_formatter)

# CORRECT: Plot the columns directly.
# The data in these columns is now normalized (0-1).
df_for_plotting[chart_columns].plot(
    ax=ax,
    kind='line',
    linestyle='-',
    linewidth=2
)

# Set plot details
plt.title('Combined Data Analysis: Trends After Min-Max Normalization',
          fontsize=16)
ax.set_xlabel('Day')
# The Y-axis now represents a relative scale from 0 to 1
ax.set_ylabel('Scaled Value (Relative Change: 0 to 1)')
ax.legend(title='Metrics')
ax.grid(True, linestyle='--', alpha=0.7)
# Set y-axis limits to clearly show the [0, 1] range
ax.set_ylim(-0.1, 1.1)

ax.set_xlabel("Date")
ax.set_ylabel("Normalized Value (0 to 1)")
# Automatically formats the dates for better readability
fig_normalized.autofmt_xdate()
```

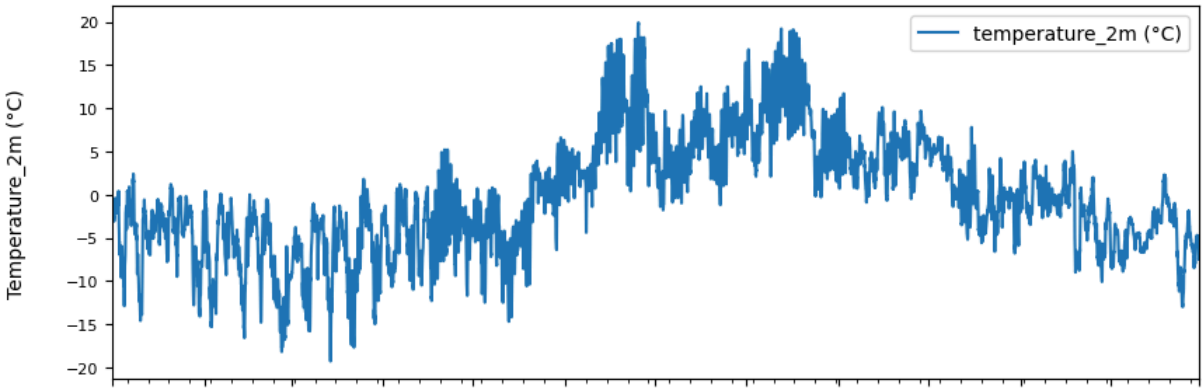
```
--- Plotting All Columns Together Using Min-Max Normalization ---
```

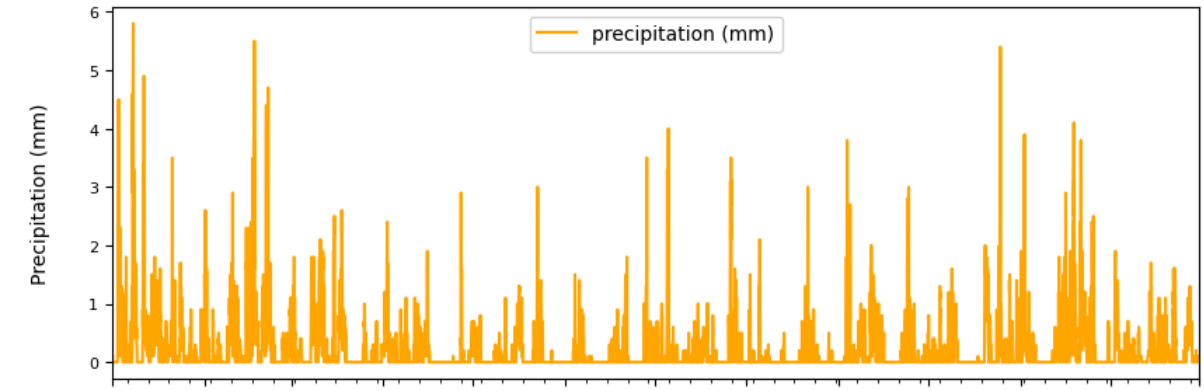


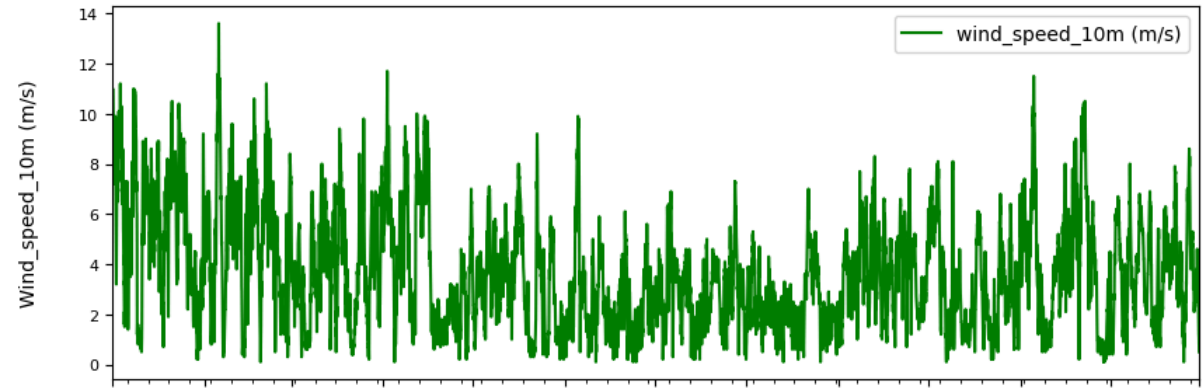
In [ ]:
```
import matplotlib.pyplot as plt
import pandas as pd # Assuming you have this imported
```

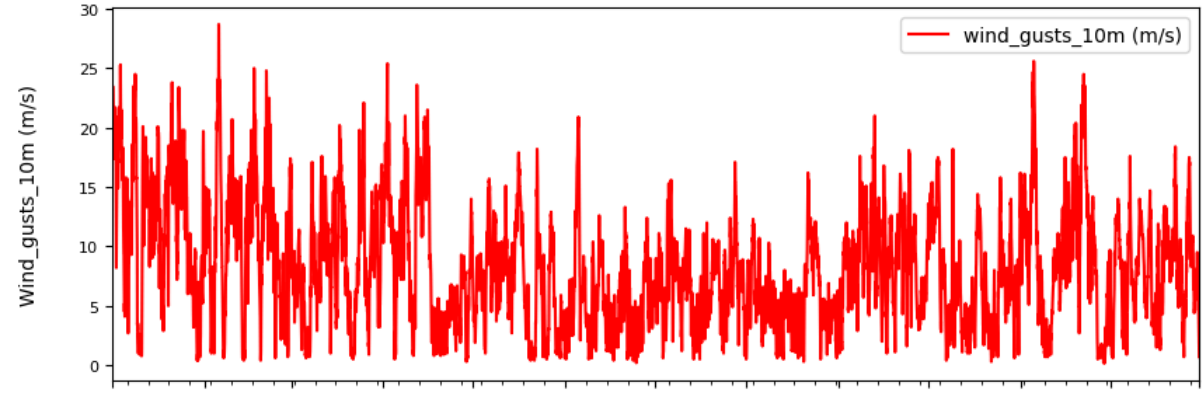
```
# 1. Set global styling defaults for all plots
plt.rc('xtick', labelsize=8) # Set font size for x-axis tick labels
plt.rc('ytick', labelsize=8) # Set font size for y-axis tick labels
plt.rc('figure', figsize=(10, 20)) # Set a default figure size

# 2. Create the figure and two subplots that share the x-axis
fig, axes = plt.subplots(nrows=5, ncols=1, sharex=True)

# 3-1. Plot on the first axes (temperature_2m (°C))
meteo_subset_DF.plot(
    x='time',
    y='temperature_2m (°C)',
    ax=axes[0],
)
axes[0].set_ylabel('Temperature_2m (°C)', rotation=90, loc='center',
                   labelpad=20)

# 3-2. Plot on the second axes (precipitation (mm))
meteo_subset_DF.plot(
    x='time',
    y='precipitation (mm)',
    ax=axes[1],
    color='orange',
)
axes[1].set_ylabel('Precipitation (mm)', rotation=90, loc='center',
                   labelpad=20)

# 3-3. Plot on the second axes (wind_speed_10m (m/s))
meteo_subset_DF.plot(
    x='time',
    y='wind_speed_10m (m/s)',
    ax=axes[2],
    color='green',
)
axes[2].set_ylabel('Wind_speed_10m (m/s)', rotation=90, loc='center',
                   labelpad=20)

# 3-4. Plot on the second axes (wind_gusts_10m (m/s))
meteo_subset_DF.plot(
    x='time',
    y='wind_gusts_10m (m/s)',
    ax=axes[3],
    color='red',
)
axes[3].set_ylabel('Wind_gusts_10m (m/s)', rotation=90, loc='center',
                   labelpad=20)

# 3-5. Plot on the second axes (wind_direction_10m (°))
meteo_subset_DF.plot(
    x='time',
    y='wind_direction_10m (°)',
    ax=axes[4],
    color='pink',
)
axes[4].set_ylabel('Wind_direction_10m (°)', rotation=90, loc='center',
```

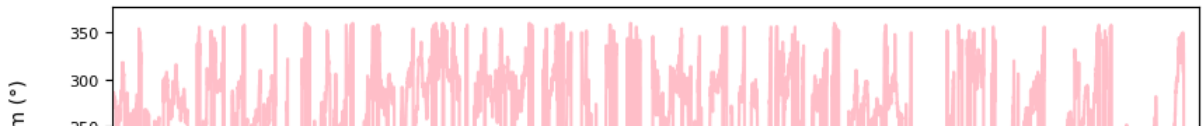
```
                    labelpad=20)

# 4. Remove horizontal space between subplots
fig.subplots_adjust(hspace=0.2)

# 5. Show the combined plot
plt.show()
```

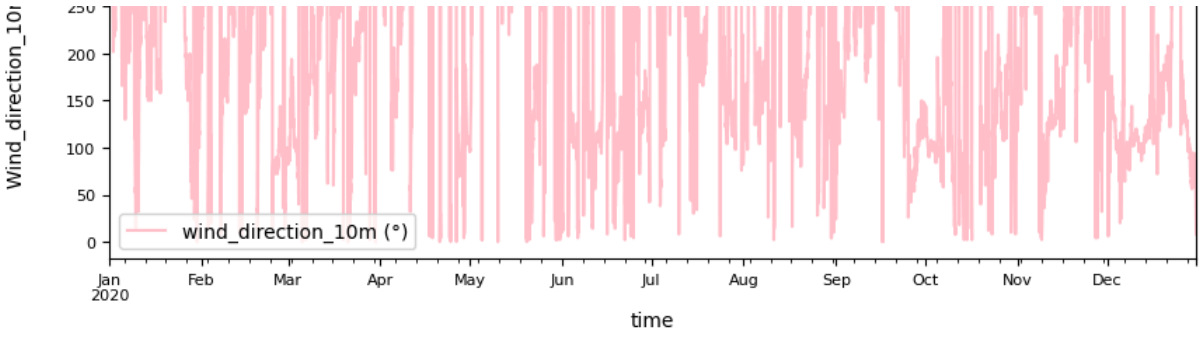

temperature_2m (°C)
Temperature_2m (°C)
20
15
10
5
0
−5
−10
−15
−20


precipitation (mm)
Precipitation (mm)
6
5
4
3
2
1
0


wind_speed_10m (m/s)
Wind_speed_10m (m/s)
14
12
10
8
6
4
2
0


wind_gusts_10m (m/s)
Wind_gusts_10m (m/s)
30
25
20
15
10
5
0


350
300



Remember to fill in the log and AI mentioned in the General section above.

Tasks, Streamlit app

Create a Streamlit app including:

- requirements.txt (for package dependencies)
- Four pages (with dummy headers and test contents for now)
- The front/home page should have a sidebar menu with navigation options to the other pages.
  -- On the second page: A table showing the imported data (see below). Use the row-wise LineChartColumn() to display the first month of the data series. There should be one row in the table for each column of the imported data. -- On the third page: A plot of the imported data (see below), including header, axis titles and other relevant formatting.
  A drop-down menu (st.selectbox) choosing any single column in the CSV or all columns together. A selection slider (st.select_slider) to select a subset of the months. Defaults should be the first month.

Data should be read from a local CSV-file (open-meteo-subset.csv, available in the Files here in Canvas), using caching for app speed.

My Streamlit app is ~~https://oriekimura-ind320-projectwork-part.streamlit.app/~~
https://oriekimura-ind320-projectwork.streamlit.app/

In [8]:
```
path_prefix = "../projects/project1/"
file_name = "Projectwork_part1.py"

# !streamlit run {path_prefix}{file_name}
```

In [ ]:
```
```