Important Information

- All formatting in this document has been applied manually.
- The formatting style is based on the guidelines from:
  IBM Jupyter Markdown Cheatsheet
- This document will include topics such as:
  - A brief description of AI usage.
  - A 300-500-word log describing the compulsory work and my experience.
  - A link to my public GitHub repository and streamlit app
  - A Table of contents with links to the place in the document
- My GitHub repository: My GitHub Repository
- My Streamlit app: My streamlit App

Table Of Contents

- Important Information
- AI usage
- My takeaway
- Bonus points

AI usage

Must include a brief description of AI usage.

In this full project over the course of 4 different compulsory assignments there are instances where AI is used.
These instances are - **Looking up errors**, **Re-writing text & documentation**.

- **Looking up Errors:** While coding you will get errors you don't understand completely, the same case is for me. Sometimes I get code errors that I don't understand, going to AI to explain the code errors for me helps me to fix the errors. Sometimes it gave me suggestions which I took, sometimes it gave me suggestions I did not like and then I looked up solutions on the web.

- **Re-writing text & documentation:** Documenting is difficult, I wrote my own takeaways and explenations for the page contents. After I wrote these I asked AI for spelling errors and sentences structure. It reprashed my words and sometimes used different grammer. I did write my own opinion and content first, as you can see because sometimes there may be spelling errors I missed..

My takeaway

Must include a 300-500-word log describing the compulsory work (including both Jupyter Notebook and Streamlit experience).

- main takeaway from this project is that I was able to build on my prior knowledge and put it into practice. I already had some experience with Python, data analysis, and visualization, but this was the first time I had to bring all those skills together into a working Streamlit app. I learned how to apply methods I had previously only tested in notebooks in a more structured and user-facing context. That felt like a real step forward, because it showed me how data work can be packaged and shared in an interactive way.

- Working with Jupyter Notebook has become a day-to-day practice for me in the past couple of months. I use it frequently for assignments and experiments, and it was interesting to see how those skills carried over. In particular, I gained more confidence in formatting notebooks with Markdown and styles. For example, before this project I did not know that I could change the background color of Markdown boxes to make sections stand out visually. That might seem like a small detail, but it makes results easier to read and present, which is valuable for both myself and any audience.

- Working with Streamlit was not entirely new to me either, but making a whole app from scratch was quite fun. It gave me more insight into how to break down a problem: first designing the page structure, then adding components like tables and plots, and finally refining the user controls. Creating the plots and tables was especially interesting. I had to combine multiple new ideas: filtering the data by month, using selection sliders, and adding line charts to a table. None of these were things I had done before, so I had to research the documentation, watch tutorials, and experiment with different approaches. The final result was satisfying, and I feel I learned methods that I can reuse in future projects.

- One specific challenge I faced was with MongoDB. MongoDB is completely new for me and this took a while to get used to and connect. After I connected Mongo everything went way smoother.

- Pipeline structure. The Medaillon pipeline structure was new for me, but this helped me structure my work way better. Splitting the retrieval of the data, cleaning and modeling in seperate files makes a way more clean working structure. After I found this out I switched most of my work to this structure which helped me to make my work faster and better.

Bonus points for compulsory 4

Below is an overview of the **bonus content** I implemented in the Streamlit application, beyond the compulsory requirements. These additions were chosen to enhance usability, robustness, and analytical depth, while ensuring the user has more control and insight into the data.

--- This block has been re-writen by AI to make it more clear what has been done ---

Waiting time enhancements (Spinners & Caching)

To improve the responsiveness of the app, I added spinners (`st.spinner`) to all time-consuming operations:

- MongoDB reads
- ERA5/Open-Meteo API downloads
- Snow drift computations
- SARIMAX model fitting

I also applied `@st.cache_data` to data-loading functions to avoid unnecessary recomputation. This drastically reduced waiting times when navigating between pages or reusing the same parameters.

Error handling (Robust UI Feedback)

Throughout the Streamlit pages, I implemented structured error handling using `try/except` blocks.
Examples include:

- Missing or misconfigured MongoDB connections
- Missing GeoJSON or Gold-layer CSVs
- ERA5 API failures
- Empty or misaligned datasets after filtering
- Invalid exogenous-variable combinations during SARIMAX forecasting

Instead of crashing, the app gives clear feedback using `st.error()` or `st.warning()`, helping the user diagnose issues quickly.

Snow Drift Bonus – Monthly Qt

In addition to the mandatory yearly snow-drift calculation (July–June seasons), I implemented:

- **Monthly Qt computation** within each season
- A **grouped bar chart** of monthly drift values by season
- An expandable table containing monthly Qt with metadata
- Integration alongside the yearly Qt analysis and wind rose

This gives a finer-grained view of snow-drift dynamics and allows comparison of seasonal patterns.

Forecasting Bonus – Exogenous Weather Variables in SARIMAX

To extend the forecasting module, I added support for **meteorological exogenous features**, including:

- Temperature
- Precipitation
- Windspeed
- Wind gusts

The implementation includes:

- Downloading ERA5 weather for the selected training period
- Aligning and cleaning weather data to match hourly/daily aggregation
- Handling ERA5 archive limits by repeating the final observed exogenous values into the forecast horizon
- Automatic dropping of incomplete rows to avoid NaN errors
- Clear warnings to inform the user when the effective forecast horizon is adjusted

This allows the SARIMAX model to incorporate weather-driven dynamics and satisfies the bonus requirement for forecasting improvements.

Summary

Bonus Category	Completed?
Waiting time (spinners + caching)	✅

Bonus Category	Completed?
Error handling & graceful UI messages	✅
Monthly snow drift	✅
Weather as exogenous variables (SARIMAX)	✅
Municipality overlay on map	❌ *(optional)*
Elevation layer	❌ *(optional)*

I implemented **four substantial bonus features**, far exceeding the requirement of one, and each of them meaningfully extends the analytical and interactive capabilities of the project.