Data Lake Engineering

Spring 2025

Final Project

Sam Neuschwander

**The Dataset**

I found the dataset that I chose to work with on the AWS Open Data marketplace. It is NOAA Global Forecast System (GFS) netCDF Formatted Data. This dataset has over 150 atmospheric and land soil variables1 and contains data from across the world. The data is formatted in NetCDF. I was unfamiliar with this format, but the information on the NOAA dataset gave a brief introduction to the format. One characteristic that I learned from the descriptions section of the dataset was that NetCDF data is ‘self-describing’.1 This is to saw that the variables contain definitions (albeit brief ones). The reason I found this beneficial is because I was finding difficulty locating a data dictionary for this dataset. I even reached out to the data owners. They provided some references that provided about the same level of detail as the self-describing netCDF files, however, were much less organized. For this reason, I decided to use the ‘Long Name’ in the variable metadata to understand what the variables represented. The data is located in an s3 bucket.2 The bucket contains daily folders that contain hourly data files. There is a three-dimensional file that is seven GBs in size and a two-dimensional file that is one GB in size for each hour of the day. The dataset starts at March 23rd, 2021, and continues to present day with new files being added every day. This amounts to nearly 300 TBs of data. A bit more than we could for the purposes of this project.

What interested me most about this project was I felt like I cloud relate it to a project I did for my feature engineering class. In this class I did a project related to California wildfires. We used weather data to understand weather patterns related to wildfires. The issue was, we were working with limited data and infrastructure, and we could only find a suitable dataset with temperature data at the state level, which is extremely unreliable when working with localized data. I thought I could marry this dataset with the wildfire data to get more accurate data, which would allow me to do some machine learning classification on this NOAA data related to wildfires. Well, it turns out, I bit off more than I could chew. The big data engineering portions of the project took me longer than anticipated, and I didn’t have time to merge any other datasets. Because of this, I decided to pivot. Instead of looking at California data, I decided to make it more local to me. I chose to analyze the Minnesota weather data and find interesting insights about weather related to where I live.

**Data Preparation**

To understand the data better I wanted to look at a couple of the datafiles individually. I wanted to dig right in, so leaning on what I know, my first exploration of the files was done in Google Colab notebooks. This gave me a quick way to understand the data a little better since I am more familiar with the platform. The data exploration gave me an idea of the size and structure of the data, understand the variables better, and understand what differentiated the files. Before I dug into the files, I didn’t know what made one file different from the next. When I looked at the time variable, I realized each file represented an hour in the day. This gave me the plans that it would be easiest to pick one file from each day, which would represent the same time every day – 1pm UTC (8AM CST).3

I spent a lot of time going over how I would bring the data in and work with it. I am less comfortable in CLI, so my general idea was to move the data I wanted to work with into a personal s3 bucket via a Cloud9 environment, then use Databricks to do most of my data preparation and manipulation. I went through several different ideas for how I wanted to bring the data in. I originally wanted to make it as automated as possible, and bring only the data I needed for a timeframe I could specify and a global location I could specify into my s3 bucket. I was going to use a python script to do this. However, I kept going back to wanting to keep my CLI code as simple as I could and do more of the work in Databricks, which felt more comfortable. I decided I was just going to bring in one full file per day for the month of January 2024 by writing this directly into my shell code. I would then filter the location data in Databricks. This ended up benefiting me, because I changed where I wanted to do the analysis from California to Minnesota, so I had all the data necessary to make that change already in my s3 bucket.

The next issue I ran into was security networking. This is a weakness for me and ended up causing me some trouble. I tried making an AIM User that I could use the credentials for in Databricks, but AWS community addition does not allow for AIM user creation. I tried creating a secure key but was having trouble attaching to my s3 bucket. In the end, the security issue had caused me enough problems that I decided to make it a public s3 bucket by attaching a public policy in my shell code and noting to delete the bucket once the data is loaded into Databricks. I know this is not an optimal solution, but it did allow me to easily access the data in Databricks.

The data I ended up moving into my s3 bucket was the 1pm two-dimensional file for each day in January 2024. This was approximately 30 GBs of data. My next step was to filter down the data I wanted to use by location in Databricks. I used a ‘for loop’ and a bounding box to only bring in data for the latitude and longitude that encompassed Minnesota, reducing the data size by quite a bit. In the ‘for loop’, I was also able to change the data file type, so I could more easily working with in in Pyspark, as spark has some trouble working with NetCDF files (thanks for the tip professor). For this reason, I changed the file type to a pandas dataframe, then a spark dataframe, and finally saved it as a parquet file, since we would be doing analytical processing on the data. This ‘for loop’ would take a while, and Databricks would terminate the cluster before the job was done. So, it was important for my ‘for loop’ to print which files were created and saved successfully. I could then go back and change the dates for my ‘for loop’ to access to pick up and process the remaining files. Once this process was complete, I was ready to analyze the data

**Data Analysis and Visualization**

I created a separate Databricks notebook to handle my data transformations and analysis. To start I merged all the filtered files. I then reviewed the size of the merged file and compared it to the size of a single filtered file to make sure it looked like we merged all 31 files successfully. I was now ready to transform, analyze and visualize the data. I chose to use the built-in data viz functions in Databricks. Getting used to some quirks took a little time (don’t click the drop down for the table when you run the visualizations, this kept crashing the cell), but it has some easy-to-use map functions. Many questions I wanted to answer were about weather events in and around Minnesota. What I noticed is because I was using a latitude and longitude bounding box, it included big sections of the surrounding states, especially western Wisconsin because the arrowhead extends so far east. I was looking for the extremes of weather events for our timeframe, and often they occurred outside of Minnesota! The highest snow cover was in two areas of Wisconsin, the highest wind was in Lake Superior, the coldest temperatures were on the South Dakota boarder and Wisconsin boarder:

A map with a location pin

AI-generated content may be incorrect.A map with a location

AI-generated content may be incorrect.

A map with blue pins

AI-generated content may be incorrect.

It was also interesting to see which variables were correlated. One that was interesting was a strong negative correlation between wind speed and 2-meter temperature (-0.84) :

A screenshot of a graph

AI-generated content may be incorrect.

**Overall Reflection**

I felt like I learned a lot when it comes to resource utilization, and the most efficient ways to work with large data. With more time, I would follow my initial plan, to tie this data to wildfire data, then run a classification model on the data to create a model that can predict wildfires. There were several challenges I encountered with this product. Longitude was represented in 360-degree terms. Often it is represented as plus minus for east and west. My first Databricks data filtering brought me data from somewhere in China, so that was a learning experience for me. Security – As discussed, I had a very difficult time figuring out the best way to secure my s3 container, and access it in Databricks. I couldn’t create an AIM User, but I know I could have created a secret key or maybe restrict access to my personal IP Address. I just had a lot of difficulty figuring out how to set this up, even though we did so in our labs. In our labs it seemed like the secret key was usually used when spinning up an EMR, which I didn’t want to do for this project. I had difficulty understanding how to apply the concept to just an s3 bucket. If I did it over again, I would do more work to figure out the security aspect of the project. Databricks Community addition timing out – This wasn’t a huge issue, but my filtering by longitude and latitude in Databricks took a long time. After an hour the notebook would time out. If I did it again, I think I would try to put this step in my shell code by calling some python code. This could present its own issues if I wanted to change the location of where I wanted to do the analysis (which I did end up doing) but I think this might have been a more efficient use of resources.

# Bibliography

1

2025. *NOAA Global Forecast System (GFS) netCDF Formatted Data.* “AWS Marketplace: NOAA Global Forecast System (GFS) NetCDF Formatted Data.” 2019. Amazon.com. 2019. https://aws.amazon.com/marketplace/pp/prodview-u4bxtndbzw4hk?sr=0-4&ref\_=beagle&applicationId=AWSMPContessa#overview.

**2** “AWS S3 Explorer.” 2025. Amazonaws.com. 2025. <https://noaa-oar-arl-nacc-pds.s3.amazonaws.com/index.html#inputs/>.

**3** “UTC to CST Converter - FreeConvert.com.” 2025. FreeConvert. 2025. https://www.freeconvert.com/time/utc-to-cst.

**‌**

**‌**