data-report

June 6, 2024

1 Data report

Yannick Rudolf, 22722156

1.1 Question

How have the number and types of weather phenomena in Germany changed over time?

1.2 Data sources

To answer the question, the historical data sets of 998 different stations of the German Weather Service are loaded. The data of each station is provided in a structured text file. The text files contain comma separated values with ";" as delimiter.

The station specific data sets contain information on the number of different weather phenomena on an annual basis. The weather phenomena considered are

Parameter	Parameter description	Data type
JA_GEWITTER	Number of days with thunderstorms per year	NUMBER
JA_GLATTEIS	Number of days with black ice per year	NUMBER
JA_GRAUPEL	Number of days with sleet per year	NUMBER
JA_HAGEL	Number of days with hail per year	NUMBER
JA_NEBEL	Number of days with fog per year	NUMBER
JA_STURM_6	Number of days with Storm > 6 on the Beaufort scale per year	NUMBER
JA_STURM_8	Number of days with Storm > 8 on the Beaufort scale per year	NUMBER
JA_TAU	Number of days with Dew per year	NUMBER

The meta data for the stations is summarized in an extra data set. The meta data is provided as a semi-structured table within a text file. The meta data contains information about the id, the name, the location, and the duration for which the respective station has recorded data.

Source: Deutscher Wetterdienst

1.2.1 Data quality

Since the German Weather Service is a state institution and the data is the state of natural events in the past, it can be assumed that the data is correct. Although the data might be correct it is not complete as there are missing values for several years on different weather stations. The 998 weather stations provide 34 years of data on average where inbetween 2.3 years are missing on average. As the weather stations are distributed all over germany the data is representative for the historical weather phenomena in germany.

1.2.2 License

The data used in this project is provided on the open data portal, Climate Data Center (CDC) through the German Weather Service. - Terms of use - License

The data is under the **Creative Commons BY 4.0 "CC BY 4.0"** License. This means the data is free to share and adapt for any purpose under the following constraints - apropriate credit to the authors - indicate if changes were made - provide a link to the license - You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

To fulfill the obligations of the license the authors of the data get mentioned in the README.md of the project, in this report, and the final report.

1.3 Data pipeline

The whole pipeline is in python and is structured as follows. Getting the data -> Processing the data to two dataframes (meta and usage data) -> Transform data -> Store data

1.3.1 Getting the data

The pipeline starts with a GET request to the provided url. Then with xpath and the lxml library all links from the reponse are extracted. The pipeline checks for every link if it is a zip-archive or a text file. The zip-archive indicates it is data from a weather station. The text file is meta data. Files with other suffixes are currently not handled. Each file is requested. The data is not written to files but the responses to the http requests are packed into io streams using the io package.

1.3.2 Processing the data

For the meta data each line is processed iteratively within a string stream. The values of each line are split into a list of strings. As the data only is semi-structured there is not always a clear seperation between the values. The header row is an exception. The header is split by whitespaces. The first 6 values are numerical values which do not include any whitespaces. Therefore, these values are split by whitespaces. The last two values are strings that can contain whitespaces (e.g. Arolsen-Neu Berich). These values are seperated with a regular expression that checks for 2 or more occurences of whitespaces and then splits the values.

Each list of strings is then stored in an additional list. The additional list is then converted to a pandas dataframe. The first row is set as column names.

For each station the data is processed within a Bytes stream. Therefore, the ZipFile package is used to open the zip archive. The file names within the zip archive are extracted. Each name is checked if it starts with 'produkt_'. This prefix indicates the actual data of the station. The other files within the zip archive are meta data which overlap with the previously described meta data and is therefore ignored. The actual data is then read with pandas with ";" as separator. The dataframe is stored in a list of dataframes.

When the stations data is processed the list of dataframes gets concatenated to a big dataframe as it is more efficent than appending the dataframes one by one.

1.3.3 Transforming the data

For the meta data the columns "Stations_id", "Stationshoehe", "geoBreite", "geoLaenge" are converted to numerical columns. The columns "von_datum", "bis_datum" are converted to datetime columns.

For the stations data only the columns "MESS_DATUM_BEGINN", "MESS_DATUM_ENDE" are converted to datetime.

The date columns are converted to improve readability and comperability. The "Stations_id" is converted due to comperability reasons to the staitons data column "STATIONS_ID". The other numerical columns are converted due to common sense.

1.4 Results

The meta data und data are stored in a SQLite database called project.sqlite. The data is written to the sql table with sqlalchemy and pandas. Within the database are two tables. "description" contains the meta data of the weather stations. The stations data is stored in the table "weather_phenomena".

1.5 Limitations

One big limitation with these data sets is that they are not static. It is historical data that still gets updated annually which will make the results of the final report obsolete next year. Yet the pipeline will still be able to update the database. It would have been an option to design the pipeline the way that it stores the data files locally. Not doing this was a conscious decision, as this pipeline should store as little data as necessary persistently and therefore only store the final database.