Police Reports and Driving Safety App

By Shriya, Arina & Filip

Shriya Walia

EC Utbildning

Data Science Project

2024-10

Content

Introduction

“Data scientists are involved with gathering data, massaging it into a tractable form, making it tell its story, and presenting that story to others.” – Mike Loukides, Editor, O'Reilly Media.

Data science covers a lot of ground. It is not just about AI or machine learning; it also involves sourcing, collecting, and cleaning up data, then transforming it into something meaningful. The real value of data depends on how it’s used. In different industries, data science can mean different things—sometimes it’s about creating a workflow that leads to smarter business decisions, and other times it’s about complex technical tasks like training AI models.

In the public space, data science can also be a tool for understanding the systems around us. By using sources like police and weather APIs, we can give people insights into what’s happening in their communities. Linking police reports data to weather can even add safety insights, showing how things like rain, wind, or snow affect driving safety and accident rates. This kind of information can help people make safer choices on the road and stay aware of conditions that impact them.

This thesis aims to develop a Streamlit app that visualizes police events and driving safety in Sweden, providing real-time weather insights using the Polis API and SMHI (The Swedish Meteorological and Hydrological Institute) API. The following questions will be addressed:

1. What is the purpose of the police reports and weather insights app developed in this project?
2. How was Agile methodology implemented in the project?
3. What is an ETL pipeline, and how was it utilized in this project?
4. What are the main features of the Händelser, Analys, and Väder pages in the Streamlit app?
5. How was the app deployed?
6. What were the limitations encountered with Streamlit in terms of performance?
7. What additional features could be added to enhance the app’s functionality?

Outline of thesis

The thesis commences with an introduction outlining the project's scope, followed by explanation of theoretical concepts essential for understanding the undertaken project. Subsequently, the methods employed, and the results obtained are discussed. A conclusive summary is provided, highlighting the insights received from the project, suggestions for enhancing performance, and a discussion of the challenges encountered.

1 Theory

Here we will go through the concepts used in building a police events and weather insights app.

1.1 Agile Working

This project is a collaborative effort by three students in the Data Science program at EC Utbildning: Shriya, Arina and Filip. To simulate a real-world work environment, we used Agile methods, dividing tasks among us, holding weekly meetings, and collaborating through a shared GitHub repository. Our project has three main components: *Händelser*, *Analys*, and *Väder*. Arina took on the foundation of the ETL pipeline from the Polisen API and developed the *Händelser* page on the Streamlit app. Filip focused on the *Analys* page and managed GitHub-related challenges, while Shriya designed the frontend of the Streamlit app and handled the *Väder* page, where she extracted and transformed data from the SMHI API.

1.2 API

"API stands for Application Programming Interface, where 'Application' refers to any software with a distinct function, and 'Interface' can be considered a service contract between two applications. This contract outlines how they communicate through requests and responses." (AWS, n.d.)

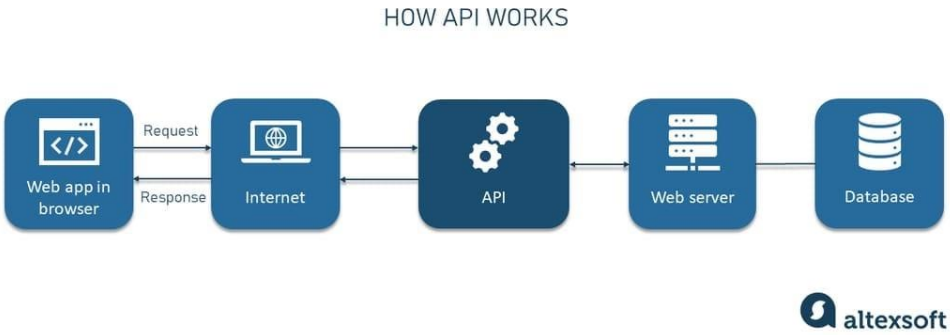

Figure 1: Working of API

- The process starts with a user opening a web app in their browser and interacting with it.
- When user selects something the app sends an API request over the internet to a specific URL tied to the API, specifying what data it needs.
- The request arrives at the API's web server, where it's interpreted.The web server acts as the middleman, checking what the user wants and gathering the necessary data.
- The web server then connects to a database that stores the requested data
- Once the database returns the data, the API organizes it into a structured format (often JSON) and sends it back over the internet to the web app.
- The web app receives the data and uses it to update the user's view.

1.3 ETL Pipeline

"An ETL pipeline is a sequence of processes designed to move data from one or multiple sources into a database, such as a data warehouse. ETL, which stands for 'extract, transform, load,' represents the three interrelated steps in data integration that enable the movement of data between databases. Once loaded, the data becomes available for reporting, analysis, and generating actionable business insights." (Snowflake, n.d.)

An ETL pipeline is used in this project where police reports data is extracted from Polisen API, transformed according to specific needs using 'Pandas' library in Python and loaded into 'Google Cloud BigQuery' for using it in the streamlit app.

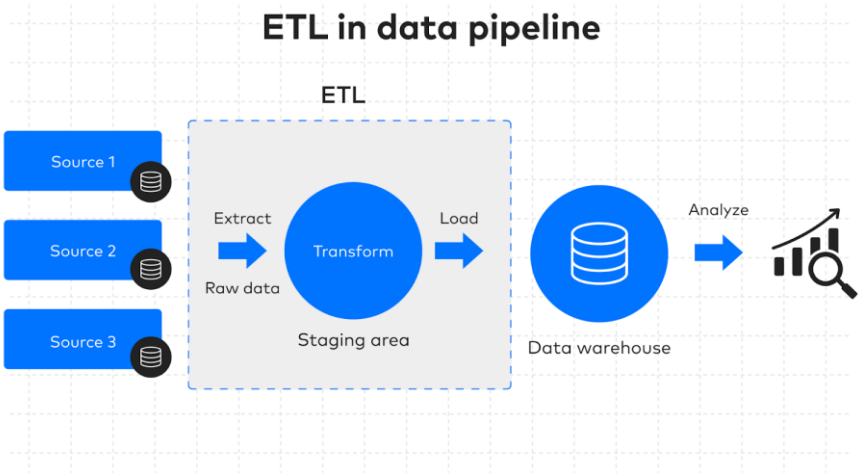

Figure 2: ETL pipleline workflow

1.4 Streamlit

"Streamlit is an open-source Python library designed to simplify the creation and sharing of custom web applications for machine learning and data science. With Streamlit, users can rapidly build and deploy robust data applications." (Snowflake, n.d.) The front end of this project is built using Streamlit, bringing the concept to life through an interactive app. Streamlit's functions, such as 'st.sidebar' (for filters and navigation), 'st.header' (for clear section headers), 'st.expander' (to hide or expand information sections), 'st.selectbox' and 'st.date_input' (for filtering by event type, location, and date), were instrumental. These tools, combined with Python, enable a user-friendly experience with dynamic content filtering, thematic structuring, and organized sidebars and tabs.

1.5 Data visualization for analysis

Visualizing data helps enhance data interpretation, making trends and insights more accessible to users. The project uses 'Folium' maps library and 'Pyecharts' library for spatial analysis of events and the statistical representation of categories, counts, and patterns over time, allowing for a more intuitive exploration of the data.

2 Method

The method includes an ETL pipeline to extract, transform, and load data from the Polisen API, while for the SMHI API, it involves only extraction and transformation. The processed data powers an interactive Streamlit app with three main components: **Händelser**, **Analys**, and **Väder**.

2.1 Tools

The backend tools used include **Python**, which served as the primary framework for building the project. Libraries like **Pandas** and **Folium** were essential for data transformation and visualization. **Google Cloud BigQuery** facilitated data storage, while **Streamlit** powered the frontend, bringing the project to life with an interactive interface.

2.2 Data Extraction

2.3 **Polisen API** : Data collection began on *September 14, 2024*, using the Polisen API endpoint: https://polisen.se/api/events, which provides the 500 most recent police reports. Notifications are usually posted within the next few hours after the incident has occurred, but this can vary depending on the situation. Sometimes the notifications are updated, when and how often this happens also varies depending on the situation. The dataset includes details such as date, time, location, latitude, longitude, event type, and event name, among other relevant information.

- **SMHI API** : Data collection from the SMHI (The Swedish Meteorological and Hydrological Institute) API is real-time, with only the latest available data used without storage. The extracted data includes date, time, temperature, precipitation, snowfall, visibility, wind gusts, wind speed, and precipitation type.

2.4 Data Transformation

Data transformation is done using 'Pandas' library on Python. Both the **händelser** and **väder** pages have different transformations.

- **Händelser och Analys (Events and Analysis):** Data extracted from the Polisen API required cleaning and transformation to enhance usability. The "name" column, which contained various report types, was consolidated into main categories for better readability. The final categories included **Trafik**, **Explosion**, **Rån och Stöld**, **Skottlossning**, **Våld**, **Brand**, and **Annat**. Additionally, the "date" and "time" were separated into individual columns and converted to a timezone-aware format in Stockholm's timezone (Europe/Stockholm), for more detailed analysis. Location information is parsed by extracting "name" and "gps coordinates" (latitude and longitude) from the "location column" into separate columns: "location_name" and "location_gps". Column data types are adjusted to appropriate formats, such as converting "id" to integer and "name", "location_name", and "summary" to strings. Unnecessary columns, like "url" are removed to keep the data relevant and streamlined.

- **Väder (Weather) :** Data extracted from SMHI API is the most recent data entry including temperature (Temperatur), wind gust (Vindby), precipitation (Nederbörd), snow precipitation (Snö), visibility (Synlighet), general wind speed (Vindhastighet), and precipitation sort code (Nederbördsortering). The “valid_time” is formatted into readable “date” and “time” strings using the ‘format_valid_time’ function. Each parameter (like temperature and wind speed) is extracted by looping through parameters to find specific weather conditions, and default values (e.g., 5 km for visibility) are set where data may be missing. The “precip_sort_code” is mapped to a descriptive label (e.g., "Snö" for snow or "Regn" for rain) for clarity. If the code does not match any key, it defaults to "Unknown."

2.5 Data Loading

Data loading is carried out for the **Händelser** and **Analys** pages, with data stored in ‘Google Cloud BigQuery’. An ETL pipeline runs daily at 10:00, collecting the latest 500 reports, transforming them, and loading them into Google Cloud BigQuery. Additionally, a Google Cloud BigQuery scheduled query runs at 11:00 each day to filter out unique events and store them in a separate table. This is necessary because the original table lacks a unique key for each event, causing previous entries to be overwritten. This setup ensures that only unique data is loaded into the app. The **Väder** page operates in real-time, so data loading isn’t needed. Instead, it fetches and displays current weather information directly on the Streamlit app without storing it, ensuring users always see the most up-to-date conditions.

2.6 Streamlit Application

The streamlit application known as ‘**Svenska Händelse-och Väderinsiktsappen**’ has 3 pages including:

- **Händelser & Analys (Events and Analysis) page**
  The **Händelser** page features a map of Sweden, with markers indicating cities where events have been reported. Each event category is represented by a distinct marker type for quick identification. Filters allow users to select by “**händelsekategori**” (event category), “**plats**” (location), and “**datum**” (date). Interactive features are created using Streamlit methods like “st.sidebar”, “st.header”, and “st.expander”, alongside Python functions for organizing the data. The **Analys** page includes the same filters and leverages Python ‘Pandas’ for data analysis, with visualizations generated using ‘Pyecharts’ to present the results.

- **Väder Sidan (Weather) page**
  The **Väder** page allows users to select a specific city and view real-time weather insights for that location via API calls to SMHI. The “**Körsäkerhet**” (Driving Safety) feature is designed to assess current weather conditions and provide a real-time indication of driving safety. This model evaluates key factors such as temperature, wind gust, precipitation, and visibility. It classifies driving conditions as "**Bra** (Good)" "**Måttlig** (Moderate)" or "**Dålig** (Poor)" based on specific thresholds: for instance, conditions are labeled "Dålig" if temperatures fall below -5°C or exceed 35°C, wind gusts are above 25 m/s, precipitation is over 5 mm, or visibility is less than 1 km.

"Måttlig" is assigned for less severe levels, while conditions are marked as "Bra" when all parameters remain within safe driving thresholds. This is done with the help of a python function using operators. It also calculates the top 5 places with the most accidents and has a special tab with "**liveuppdateringar (Live updates)"** where user can give live updates about experiences on road in a particular city.

2.7 App Deployment

The app was deployed once all components were functioning smoothly and as expected. Deployment utilized our 'GitHub' repository, where we managed the project collaboratively. A special "secret key" was generated to enable secure access to 'Google Cloud BigQuery' during deployment, and a "requirements file" was created to ensure the Streamlit app had all necessary libraries installed. This setup allowed seamless functionality and secure, real-time data access for the app.

3 Result

The results are displayed in the link (https://policeapi.streamlit.app/) and the images below. The app functions effectively, offering multiple interactive features that allow users to gain insights into both police-reported events across Sweden and current driving safety conditions.

Introduction page:

Figure 3: Intoduction page on Streamlit app

Händelser page:

Figure 4: Händelser page on Streamlit app

Analys page:

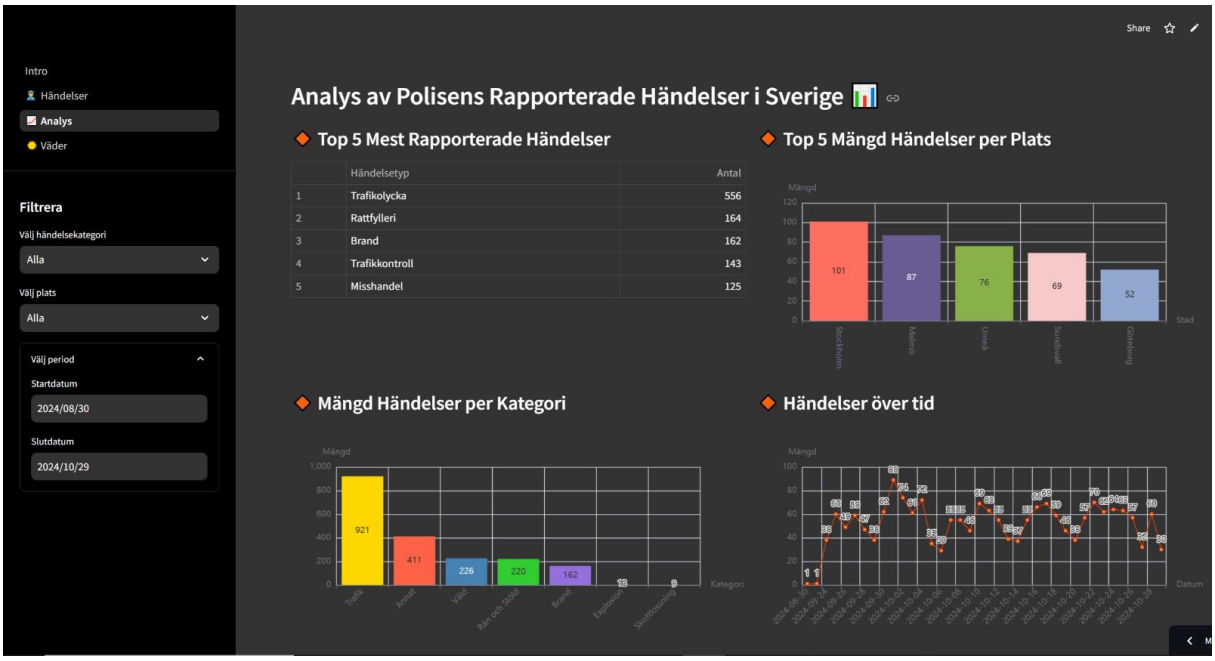

Figure 5: Händelser anlays page on Streamlit app

Väder page:

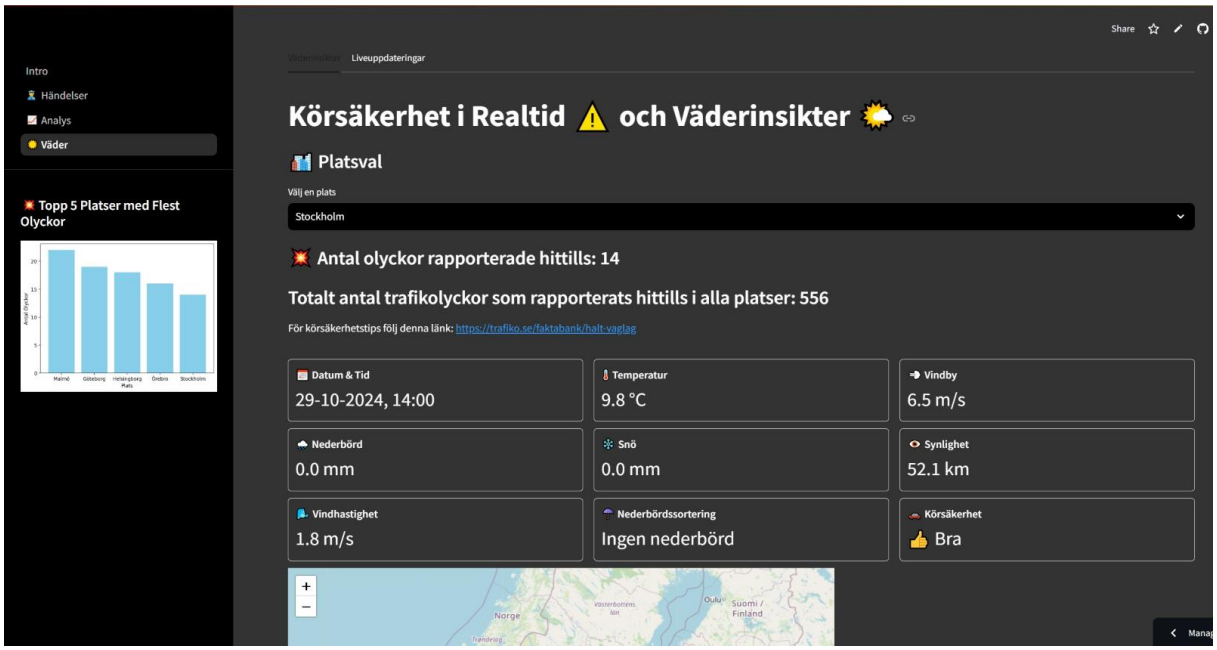

Figure 6: Körsäkerhet & Väderinsikter page on Streamlit app

4 Conclusion

4.1 Data storage and visualization

- 'Google Cloud BigQuery' is highly effective for data storage, handling large datasets with ease. The only drawback is that it does not have a feature of unique keys.
- 'Streamlit' enabled an interactive and user-friendly interface, though with some processing limitations for real-time data.

4.2 Streamlit app performance

The original plan for the *Väder* page was to display safety indicators (red, yellow, green) across all of Sweden on a map, showing real-time driving safety conditions. Due to performance issues, specifically, long loading times of about 10 minutes—the design was simplified to show weather insights for one city at a time, improving speed and usability.

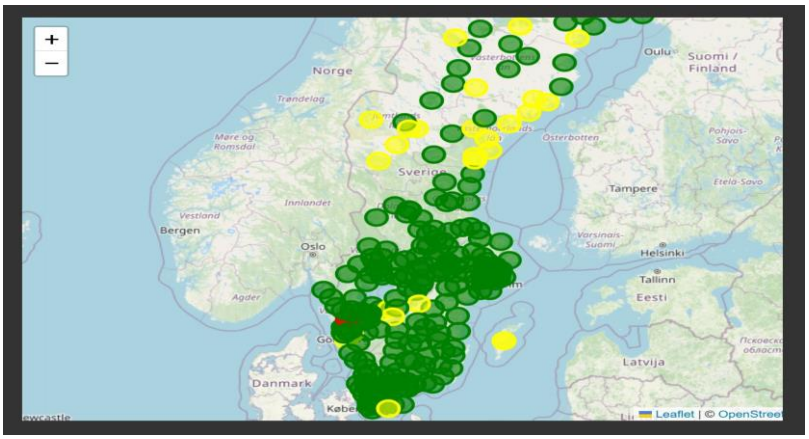

Figure 7: Original plan: show driving conditions across Sweden

4.3 Limitations of Streamlit

'Streamlit' lacks functionality for shared sidebar filters across multiple pages, unlike Power BI. As a result, the *Händelser* and *Analys* pages had to use separate filters, which reduced the interactive flow of the app.

4.4 Potential for future expansion

- With additional time, the project could have incorporated more data sources, such as social media and news articles, through web scraping, to enhance insights.
- Collecting a larger dataset over time could support the development of machine learning models, potentially enabling:
  - → Prediction of crime trends in specific areas.
  - → Assessment of accident risks based on historical weather conditions and accident correlations.

This project demonstrates promising potential for further refinement and expanded functionality, with room to grow into a predictive and more comprehensive tool.

5 References

- Amazon Web Services. (n.d.). *What is an API?*. Amazon. Retrieved from https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.
- ChatGPT. (2024)
- Fivetran. (n.d.). *Data Pipeline vs. ETL - Figure 2*. Retrieved from https://www.fivetran.com/learn/data-pipeline-vs-etl
- Snowflake. (n.d.). *What is an ETL pipeline?*. Snowflake. Retrieved from https://www.snowflake.com/guides/etl-pipeline/
- Snowflake. (n.d.). *About Streamlit*. Snowflake. Retrieved from https://docs.snowflake.com/en/developer-guide/streamlit/about-streamlit
- AltexSoft. (n.d.). *What is API: Definition, Types, Specifications, Documentation - Figure 1*. Retrieved from https://www.altexsoft.com/blog/what-is-api-definition-types-specifications-documentation/

6 Självutvärdering

1. **Utmaningar du haft under arbetet samt hur du hanterat dem.**
   → Utmaningar jag hade under projektarbetet var mest relaterade till den Streamlit appen. de andra aspekterna utfördes smidigt.

2. **Vilket betyg du anser att du skall ha och varför.**
   → Jag tror att vår grupp har visat en god förståelse för arbetsflödet inom data science, vilket speglar branschpraxis som är både relevant och värdefull att lära sig. Detta borde ge oss ett VG, eftersom vi har gjort vårt bästa för att på ett korrekt sätt representera en praktisk del av data science som används i företag idag.