Polis och Väder Applikation

Av Filip, Arina och Shriya

Filip Östlund

EC Utbildning

Polis och Väder Applikation

202410

Abstract

This project explores the integration of real-time data from public safety and weather sources to enhance driving safety insights in Sweden. Working in an agile team consisting of Arina Godman, Shriya Walia, and myself, we iteratively developed a Streamlit-based application that visualizes police-reported events throughout Sweden. The application integrates APIs from the Swedish Police and the Swedish Meteorological and Hydrological Institute (SMHI), combining police-reported incidents with current weather conditions to assess their impact on road safety.

Data is processed through an ETL pipeline, organized in BigQuery, and presented interactively to provide users with real-time visualizations. The agile approach allowed us to respond flexibly to the challenges we faced, while maintaining a collaborative and adaptive workflow. This project demonstrates the potential of data science in public safety applications. Future development could expand data sources and refine safety metrics for broader applications.

Innehållsförteckning

1 Inledning

Datavetenskap spelar en alltmer central roll i att hantera och analysera stora mängder data för att skapa en bättre förståelse av samhällsrelaterade frågor, däribland den offentliga säkerheten och väderförhållanden. Särskilt viktigt är detta i ett land som Sverige, där både väder- och säkerhetsfaktorer har en direkt påverkan på människors vardag. Genom att erbjuda realtidsdata och lättillgängliga analyser kan teknik och datavisualisering bidra till att ge allmänheten en bättre överblick över händelser relaterade till säkerhet och väder.

Detta projekt avser att utveckla en applikation där användare kan få insikter om två huvudsakliga områden: polisrapporterade händelser och trafiksäkerhet kopplat till väderförhållanden. Genom att använda polisens API kan applikationen hämta och visualisera aktuella händelser och incidenter rapporterade av polisen, vilket ger användare en tydlig överblick över den allmänna säkerhetssituationen i Sverige. Den andra delen av applikationen integrerar väderdata från SMHI för att analysera väderförhållanden i relation till vägtrafiksäkerhet, där exempelvis temperatur, nederbörd och sikt kan ha en direkt påverkan på trafiksäkerheten.

1.1 Syfte

Syftet med denna rapport är att utveckla en applikation som visualiserar realtidsdata för att förbättra förståelsen av den offentliga säkerheten och trafiksäkerheten i Sverige. För att uppnå detta syfte kommer följande frågeställningar att besvaras:

1. Hur kan aktuella polisrapporter visualiseras för att ge användare en lättöverskådlig bild av säkerhetsläget i Sverige?
2. Hur kan väderfaktorer och trafiksäkerhetsdata presenteras för att främja tryggare beslut kring resande?
3. Går det via gratis verktyg sätta upp en applikation tillgänglig för allmänheten?

Genom att besvara dessa frågor strävar rapporten efter att visa hur en datavisualiseringsapplikation kan ge handlingsbara insikter för allmänheten och därmed bidra till ett säkrare samhälle.

2 Teori

2.1 API

Ett API (Application Programming Interface) är en samling regler och protokoll som gör det möjligt för olika mjukvarusystem att kommunicera med varandra. API används ofta för att hämta data från externa källor i realtid, och de spelar en avgörande roll i dagens datavetenskap (Contentful, 2023, What is an API? How APIs work, simply explained).

Varje API består av olika ”endpoints” som kan tillhandahålla specifika dataset beroende på användarens behov. Data som hämtas från ett API behöver vanligtvis bearbetas och struktureras för att vara användbara i analys- och visualiseringsverktyg (Contentful, 2023). API i detta projekt möjliggör kontinuerlig datauppdatering, vilket gör applikationen särskilt användbar för att presentera aktuella insikter.

2.1.1 Polisens API

Polisens API över händelser innehåller 500 aktuella notiser om polisutryckningar med kortfattad information. Informationen bygger på inledande uppgifter som kan komma att ändras. Notiser publiceras vanligtvis inom några timmar efter händelsen, men tiden och eventuella uppdateringar kan variera beroende på situationen. API:et nås via https://polisen.se/api/events. Datamängden innehåller inga personuppgifter (Polisen, 2021, API över Polisens Händelser)

2.1.2 SMHI:s API

SMHI:s API för meteorologiska observationer erbjuder tillgång till öppna data om väderförhållanden och använder HTTP som kommunikationsprotokoll. API:en är uppbyggt som en REST-tjänst och nås via https://opendata-download-metobs.smhi.se/api (SMHI, 2024, SMHI Open Data API Docs)

2.2 BigQuery

BigQuery är en molnbaserad datahanteringstjänst från Google som möjliggör lagring och analys av stora datamängder. Tjänsten är särskilt effektiv för att bearbeta och analysera stora dataset snabbt och är därför populär för avancerade dataprojekt. BigQuery bygger på ett strukturerat SQL-baserat språk, vilket gör det enkelt att skriva frågor för att extrahera och sammanställa data (Kazunori Sato, 2012, Google Cloud BigQuery Docs).

Tack vare BigQuerys skalbarhet och snabbhet kan man enkelt hantera stora mängder data och köra SQL-frågor för att filtrera och analysera data i realtid. BigQuerys funktioner möjliggör också smidig integration med andra verktyg, vilket underlättar dataflödet, exempelvis i detta fall till en applikation.

2.3 Paket

2.3.1 Streamlit

Streamlit är ett ramverk för Python som gör det möjligt att enkelt skapa webbapplikationer med datafokuserade visualiseringar och interaktiva funktioner. Verktyget är särskilt uppskattat inom datavetenskap och maskininlärning, eftersom det är lätt att använda och snabbt att implementera. Streamlit fungerar genom att konvertera Python-kod till en interaktiv webbapplikation, vilket möjliggör en direkt koppling mellan analyskod och användargränssnitt. Genom Streamlits inbyggda stöd för widgets, filter och visualiseringar kan användarna snabbt navigera och interagera med de data som presenteras (Treuille, A. et al., 2018, Streamlit documentation).

2.3.2 Folium

Folium är ett Python-bibliotek för att skapa interaktiva kartor, särskilt anpassat för datavisualiseringar som kräver geografiska kartläggningar. Folium bygger på Leaflet.js, ett populärt JavaScript-bibliotek för kartor, och gör det enkelt att generera dynamiska kartor direkt i en Python-miljö. Foliums förmåga att hantera olika lager av geografisk information gör det också möjligt att skapa komplexa visualiseringar som är enkla att tolka. (Rob Story, 2024, Folium Documentation)

2.3.3 Pyecharts

Pyecharts är ett visualiseringsbibliotek för Python som möjliggör skapandet av vackra och interaktiva grafer och diagram. Pyecharts bygger på ECharts, ett JavaScript-bibliotek, vilket ger hög prestanda och flexibilitet för att skapa anpassade diagram. Pyecharts interaktiva funktioner gör att användaren kan fokusera på olika delar av diagrammen, vilket ger en mer engagerande och informativ upplevelse (Chenjian, D et al., 2024, Pyecharts Documents)

3 Metod

3.1 Arbetssätt

Projektet genomfördes enligt ett agilt arbetssätt med regelbundna möten två gånger i veckan för att följa upp framsteg, identifiera hinder och justera mål. Denna process möjliggjorde snabb anpassning och effektiv problemlösning, vilket säkerställde ett strukturerat arbetssätt.

Jag ansvarade för projektets Analyssida och Github hantering, samt allmänna buggar precis som resten av teamet. Arina skötte utvecklingen av ETL-pipelinen för polisens data samt streamlit-sidan för händelser. Shriya fokuserade på designen av hela streamlit gränssnittet och utvecklingen av sidan för väder.

3.2 Datainsamling

3.2.1 Polis API

Polisens API användes för att samla in data om rapporterade händelser och insatser från polismyndigheten. Genom detta API kunde information om aktuella händelser, såsom trafikolyckor, rån och andra säkerhetsincidenter, hämtas i realtid. Eftersom polisens API endast sparar de senaste 500 händelserna behövde vi lagra datan i en databas för att ha en historik längre bak i tiden.

3.2.2 SMHI API

Väderdata samlades in via SMHI API, som tillhandahåller meteorologiska observationer för utvalda svenska städer. API:et hämtar löpande väderinformation som temperatur, nederbörd och vindhastighet, och data uppdateras periodiskt för att återspegla aktuella väderförhållanden. Denna data uppdateras kontinuerligt och används för att analysera väderfaktorers påverkan på trafiksäkerheten. Till skillnad från polisens händelsedata sparas inte väderinformationen i en databas utan används endast för realtidsanalys.

3.2.3 ETL-Pipeline

För att effektivt samla in, bearbeta och lagra data från polisens API skapades en ETL-pipeline (Extract, Transform, Load). Pipelinens huvudsakliga syfte var att hämta rådata från Polisens API och lagra den i Google BigQuery, där data därefter transformerades för att möjliggöra analyser. Pipelinen automatiserades för att säkerställa kontinuerlig uppdatering av polisdata, så att aktuell information alltid är tillgänglig.

3.2.4 Windows Task Scheduler

För att schemalägga och automatisera hämtningen av data från API användes Windows Task Scheduler. Med hjälp av denna tjänst kunde polishändelsernas ETL-process köras, samt väderdatan kunde hämtas direkt från SMHI:s API vid fasta tidsintervall, vilket möjliggjorde kontinuerlig uppdatering av både händelse- och väderdata utan manuell inblandning.

3.3 Data Transformation

Efter insamlingen genomgick datan som tidigare nämnt i ETL-pipelinen en transformationsprocess för att säkerställa enhetliga datatyper och format. För polisdatan innebar detta att kolumntyper och datatyper justerades för att underlätta analys, medan eventuella saknade värden eller felaktigheter rättades till. Sedan i Python delades händelsetyperna från polisrapporteringen vidare in i huvudkategorier för att skapa en tydligare överblick över alla händelser, vilket gör det lättare att förstå samt senare också filtrera kring olika kategorier. Väderdatan genomgick som tidigare nämnt ingen etl-pipeline vilket betyder att all transformation gjordes i Python. Det var mest att transformera datan för att göra den enklare för användaren att förstå. Till exempel ändrades sifferkod för olika mängder regn till deskriptiva förklaringar (0 = ”Ingen nederbörd”).

3.4 Streamlit-applikation

3.4.1 Streamlit

Streamlit användes för att utveckla själva applikationen som gör det möjligt för användaren att visualisera och analysera polisrapporterade händelser och väderinformationen. Designen blev tre olika sidor en för Händelserna, en för Analys av händelserna och sist en för Väder/Trafik. För alla tre sidor finns filtreringsmöjligheter för att ge användarna en anpassad vy av datan. 80% om inte mer av applikationens design är Streamlits egna inbyggda funktioner och paket.

3.4.2 Folium

För att visualisera de polisrapporterade händelserna användes Folium som kartverktyg. Vi skapade en dynamisk karta där polisens rapporterade händelser representerades med klickbara markörer, vilket ger användaren möjlighet att se detaljerad information om varje händelse direkt på kartan. Markörerna har anpassats för att reflektera olika huvudkategorier av händelser. Om flera händelser rapporterats från samma plats, hanteras detta genom att använda en speciell markör som visar en sammanfattning av de olika händelsekategorierna. På detta sätt kan användaren snabbt identifiera och undersöka specifika incidenter i olika regioner. Folium finns både som vanligt python paket, samt ett specifikt streamlit-folium paket som också användes i detta fall.

3.4.3 Pyecharts

För att skapa interaktiva diagram och tabeller användes Pyecharts, vilket gjorde det möjligt att visualisera analyser av de mest frekvent förekommande händelserna och deras fördelning över tid, plats och kategori. Datan visualiseras av 2st interaktiva stapeldiagram, ett linjediagram samt en tabell. Dessa visualiseringar förbättrar användarens förståelse för säkerhetsläget över tid och underlättar identifiering av potentiella mönster, såsom vanliga brottstyper eller olycksrisker i specifika områden. Precis som Folium finns Pyecharts som vanligt python paket och i streamlit paket men då som Streamlit-Echarts (Echarts är pyecharts bas).

4 Resultat och Diskussion

Resultat blev en fullt fungerande streamlit applikation som nås via denna länk: https://policeapi.streamlit.app/. Nedan följer hur resultatet blev för varje sida, förutom Intro sidan då den endast innehåller en kortare text med info om projektet.

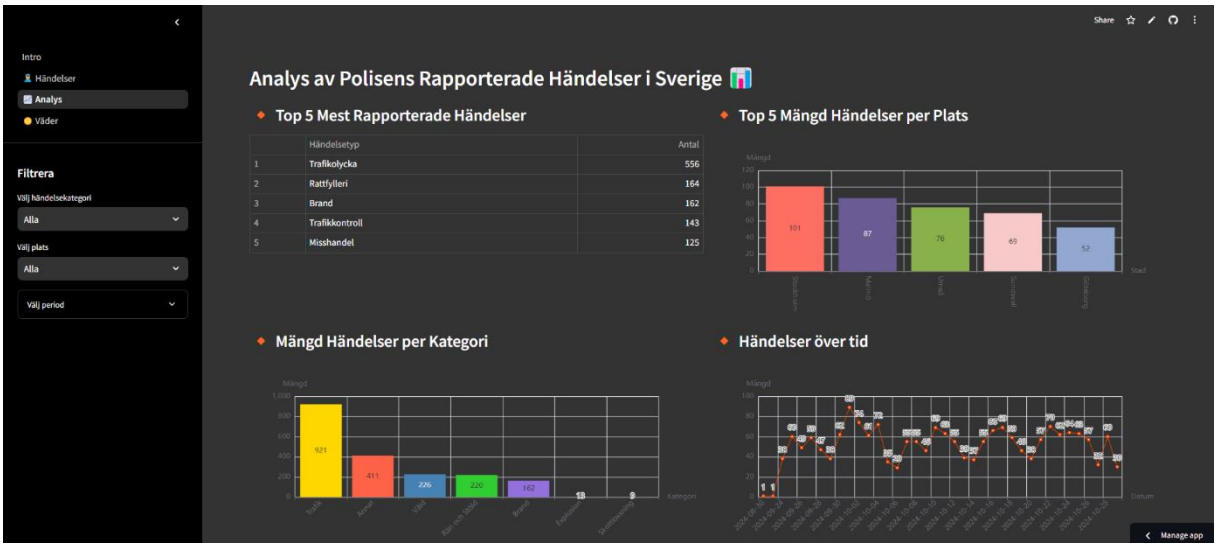
4.1 Sida 1 – Händelser

Figur 1 - Skärmavbild från applikationens Händelse-sida

På Händelser sidan visualiseras polisanmälda händelser runtom i Sverige med hjälp av en interaktiv karta. Varje händelse är markerad med en ikon som representerar en specifik händelsetyp, såsom trafikolyckor, inbrott, bränder och andra typer av incidenter. Användaren kan filtrera händelserna baserat på kategori, plats och tidsperiod, vilket gör det möjligt att anpassa kartan efter specifika intressen eller behov.

Vid klick på en markör visas detaljerad information om den valda händelsen i en panel till vänster, inklusive län, händelsetyp, datum och tid, samt en kort beskrivning. Bilden visar ett exempel där en trafikincident i Jokkmokk visas, med information om tidpunkt och detaljer kring händelsen. Detta ger användaren snabb och enkel tillgång till aktuell och relevant information om polisanmälda händelser i realtid.

4.2 Sida 2 – Analys

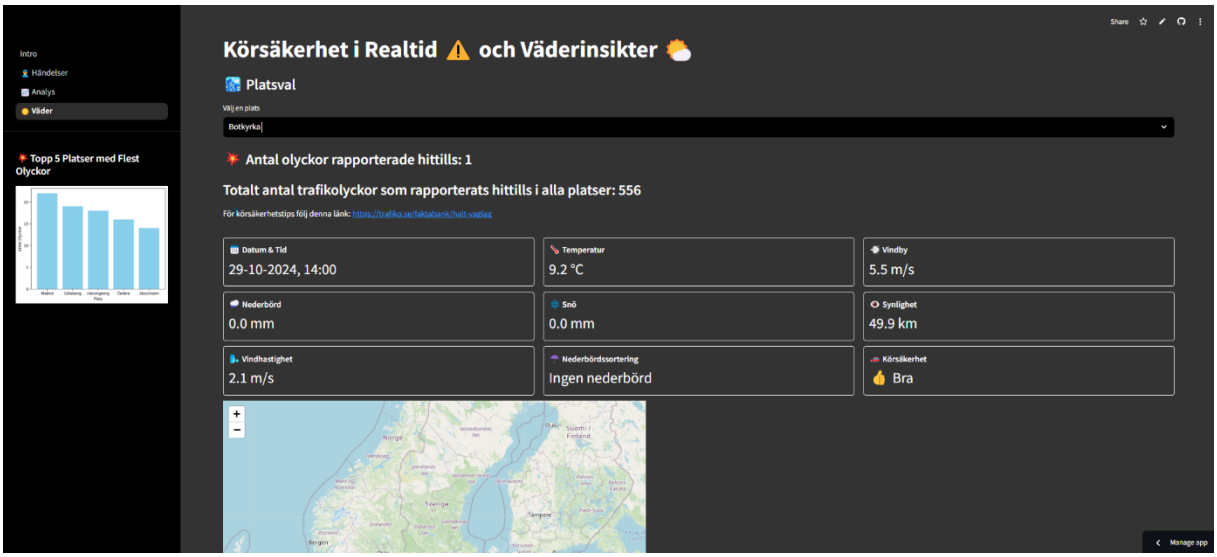


Figur 2 - Skärmavbild på applikationens Analys-sida

På Analys sidan presenteras en fördjupad analys av de polisanmälda händelserna i Sverige med hjälp av interaktiva visualiseringar. Grafiken är uppdelad i flera sektioner för att ge en översikt över mönster och trender i de rapporterade incidenterna. Första sektionen visar top 5 mest rapporterade händelser i tabellformat. Andra sektionen visar top 5 mängd händelser per plats i stapelformat. Tredje sektionen visas mängd händelser per kategori också i stapelformat. I sista sektionen visas händelser över tid i linjeformat.

Sidan ger användaren möjlighet att filtrera datan utifrån händelsetyp, plats och tidsperiod, vilket gör det enklare för användaren att anpassa visualiseringarna efter specifika behov. Eftersom pyecharts grafer är interaktiva kommer de ändras och om man till exempel väljer Stockholm kommer den platsens händelser endast presenteras i alla grafer. På bilden ser vi alla händelser på alla platser.

4.3 Sida 3 – Väder och Trafiksäkerhet



Figur 3 - Skärmavbild på applikationens Väder-sida

Vädersidan ger en överblick av trafikolyckor relaterade till väderförhållanden i realtid, vilket är avgörande för att förstå körsäkerheten i specifika områden. Användare kan välja en plats som i bildens exempel är Botkyrka, och få en översikt av aktuella väderdata samt körsäkerhetsinformation för området. Sidan innehåller mycket info, nedan är sidans info sammanfattad:

- **Trafikolyckor:** Sidan visar antalet rapporterade trafikolyckor i valt område samt det totala antalet olyckor som rapporterats i hela landet.
- **Väderdata och Körsäkerhet:** Information om väderförhållanden som temperatur, vindhastighet, vindbyar, nederbörd, snömängd, och synlighet presenteras tydligt.
- **Körsäkerhetsstatus:** En indikator anger om körsäkerhetsförhållandena är "Bra" eller potentiellt riskfyllda. Detta hjälper användaren att snabbt bedöma trafiksäkerheten.
- **Topp 5 Platser med Flest Olyckor:** En graf i vänsterkolumnen visar de fem städer där flest trafikolyckor har rapporterats.

4.4 Utmaningar

Applikationen uppfyller de målen vi satt och ger användarna en effektiv lösning för att följa polisär information och väderförhållanden i realtid. Under utvecklingen identifierades dock vissa utmaningar som ledde till vissa ändringar till den slutliga produkten.

4.4.1 Streamlit synkronisering

För att förbättra användarvänligheten försökte vi initialt samla Händelser- och Analyssidan i en sida med flikar i stället. Målet var att filtrering på en flik skulle gälla för den andra. Tyvärr stöder Streamlit inte synkroniserad uppdatering mellan flikar, vilket gjorde att filterinställningar och information inte överfördes automatiskt. För att säkerställa en stabil användarupplevelse delades därför funktionerna upp på två sidor.

4.4.2 BigQuery Primary Key

BigQuery möjliggjorde lagring av stora datamängder men saknade stöd för primary key constraints, vilket försvårade lagringen av unika händelser. Detta löstes genom en separat tabell för endast de unika incidenterna som skapades genom en daglig SQL-fråga. Detta ökade lagringsbehov men eliminerade redundanta data.

4.4.3 API prestanda

Vid lokal utveckling på våra egna datorer kunde användare välja alla städer samtidigt på vädersidan för en nationell överblick. När applikationen laddades upp på Streamlit visade sig dock dess beräkningsresurser vara begränsade, vilket resulterade i långa laddningstider, särskilt för data från SMHI-API. För att öka prestandan och den generella användarupplevelsen valde vi att visa väder och trafikdata endast för en plats i taget.

5 Slutsatser

5.1 Fråga 1

Hur kan aktuella polisrapporter visualiseras för att ge användare en lättöverskådlig bild av säkerhetsläget i Sverige?

Projektet visar att det är möjligt att skapa en lättöverskådlig och användarvänlig visualisering av säkerhetsläget i Sverige genom att använda aktuella polisrapporter. Genom en interaktiv karta, där användare kan filtrera information efter händelsetyp, plats och tidsperiod, ges en tydlig överblick över händelser runt om i landet. Denna typ av visualisering gör det enkelt för användare att snabbt få insikt i säkerhetsläget i olika områden och identifiera potentiella risker, vilket kan öka medvetenheten och trygghetskänslan.

5.2 Fråga 2

Hur kan väderfaktorer och trafiksäkerhetsdata presenteras för att främja tryggare beslut kring resande?

För att främja säkrare beslut kring resande har även väder- och trafiksäkerhetsdata integrerats i applikationen. Genom att visualisera faktorer som temperatur, nederbörd, sikt och vindhastighet i realtid i relation till trafiksäkerhet kan användare enkelt få en överblick av vägförhållanden innan de ger sig ut. Denna information hjälper resenärer att förstå hur olika väderförhållanden kan påverka deras resa och ger möjlighet att planera mer säkert utifrån aktuella och potentiella risker på vägarna.

5.3 Fråga 3

Går det via gratis verktyg sätta upp en applikation tillgänglig för allmänheten?

Slutligen visar projektet att det är fullt möjligt att sätta upp en applikation tillgänglig för allmänheten med hjälp av gratis verktyg. Genom att använda ramverk och bibliotek som Streamlit för front-end-utveckling, Folium för kartvisualisering, och BigQuery för datalagring, har applikationen byggts utan kostnadskrävande verktyg eller tjänster. Detta gör lösningen både tillgänglig och skalbar till en viss gräns, vilket är värdefullt för att sprida samhällsinformation på ett kostnadseffektivt sätt.

5.4 Förbättringar

Som med de flesta skolprojekt finns det potential för vidare utveckling och förbättringar. Ökad kapacitet för hantering av samtidiga API-anrop skulle kunna möjliggöra mer omfattande och frekvent datavisualisering utan att påverka applikationens prestanda. Exempelvis skulle en lösning för snabbare och parallella API-anrop kunna minska laddningstiden för väder- och trafikdata, vilket förbättrar användarupplevelsen vid realtidsanalyser.

Vidare skulle fler och mer flexibla filterfunktioner kunna implementeras för att ge användare möjlighet till en mer detaljerad analys av säkerhetsdata. Genom att tillåta anpassade vyer och historiska jämförelser skulle användare få djupare insikter och en bättre förståelse för trender inom både väder- och säkerhetsområden.

Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

   Några utmaningar som vi hade nämndes i rapporten. Utöver det så kanske det kunde vara uppdelningen i början. Vem skulle göra vad, men det löste sig ganska snabbt och vi kom in i ett bra flow då vi träffades två gånger i veckan och samtalade i vår chatt under hela projektet.

2. Vilket betyg du anser att du skall ha och varför

   Jag tycker alla i gruppen ska ha VG då alla bidragit och förstått hur man samarbetar i grupp. Samt lärt sig använda agila arbetsätt.

3. Något du vill lyfta fram till Antonio?

   Rolig kurs, hoppas vi syns i vår!

Källförteckning

Contentful, (2023) What is an API? How APIs work, simply explained. https://www.contentful.com/api/, Hämtad 2024-10-24.

Polisen, (2021) API över Polisens Händelser. https://polisen.se/om-polisen/om-webbplatsen/oppna-data/api-over-polisens-handelser/, Hämtad 2024-10-24.

SMHI, (2024), SMHI Open Data API Docs. https://opendata.smhi.se/apidocs/metobs/index.html, Hämtad 2024-10-24.

Kazunori Sato, (2012) Google Cloud BigQuery Docs. https://cloud.google.com/bigquery/docs/introduction, Hämtad 2024-10-25.

Treuille, A. et al., (2018) Streamlit documentation, https://docs.streamlit.io/, Hämtad 2024-10-25.

Rob Story, (2024) Folium Documentation, https://python-visualization.github.io/folium/latest/, Hämtad 2024-10-28.

Chenjian, D et al., (2024) Pyecharts Documents, https://pyecharts.org/#/en-us/, Hämtad 2024-10-28.