KLIMADATENCHALLENGE

Team Sturm

Adem Cutura, Benjamin Würmli und Silas Tschopp

Inhalt

Dokumentation Klimadatenchallenge Team Sturm 2024

Das Team

Das Klimadatenchallenge-Team "Sturm" besteht aus drei Personen mit unterschiedlichen Hintergründen und Kenntnissen.

Adem Cutura ist Werkstudent im Bereich Finanzen bei der Evonik International AG. Er hat zwei Jahre Data Science an der Ludwig-Maximilians-Universität (LMU) in München studiert, bevor er an die FHNW wechselte. In seinem aktuellen Beruf ist er es gewohnt, Präsentationen zu erstellen. In dieser Challenge übernahm Adem die Erstellung der Präsentationen für die In-Flight Workshops und die Entwicklung der Datastory.

Benjamin Würmli arbeitet als Informatiker bei der Bertschi AG und bringt umfangreiche IT-Kenntnisse ins Team ein, die im späteren Verlauf der Challenge sehr nützlich sein werden. Da Benjamin ein Semester weiter ist als Adem und Silas, trägt dies ebenfalls zur Stärke des Teams bei. Aufgrund seiner umfassenden Programmiererfahrung übernahm Benjamin die Leitung bei der Erstellung des Dashboards.

Silas Tschopp ist gelernter Automatiker und arbeitet bei der Actemium Schweiz AG. Durch seine Erfahrung im Projektmanagement kennt er die wesentlichen Aspekte, auf die man bei einem Projekt achten muss. Deshalb übernahm er die Rolle des Projektleiters während dieser Challenge. Zusätzlich unterstützte er Adem und Benjamin bei der Erfüllung ihrer Aufgaben.

Datenauswahl

Bei der Suche nach geeigneten Daten für unsere Challenge war es uns wichtig, dass der Datensatz möglichst viele Städte in Europa abdeckt und die Daten, wenn möglich, bis ins Jahr 1980 zurückreichen. Nach einer umfassenden Internetrecherche stiessen wir auf die Seite der «Historical Weather API» von Open-Meteo (open-Meteo, 2024). Diese Datenquelle bietet eine breite Palette von Wetterdaten, die bis ins Jahr 1940 zurückreichen. Neben unserem Fokus auf Sturmdaten können auch Informationen zu Temperaturen, Niederschlag, Luftfeuchtigkeit, Bodentemperaturen und vielem mehr abgerufen werden.
Ein grosser Vorteil dieser Datenquelle für unsere Challenge war die Verfügbarkeit eines Python-Programms zur Abfrage der Application Programming Interface (API). Durch geringfügige Anpassungen des Programms konnten wir die benötigten Winddaten von der API abrufen. Ein weiterer bedeutender Vorteil dieser API ist die kontinuierliche Aktualisierung der Daten. Das bedeutet, dass wir ab dem Tag, an dem wir begannen mit den Daten in unserem Programm zu hantieren, keine Unterbrechung der Datenversorgung haben und stets die neuesten Winddaten einsehen können.
Die einfache Handhabung und die Vielfalt der verfügbaren Daten machen die «Historical Weather API» von Open-Meteo zu einer ausgezeichneten Wahl für unsere Challenge. Sie ermöglicht es uns, umfassende historische Wetteranalysen durchzuführen, welche uns bei der Auswahl der Datenstory wesentlich unterstützte.

Daten Aggregierung

In unserem Dashboard benötigen wir die Aggregation der Daten auf Monats- und Jahresebene. Der Benutzer soll auf einen Blick erkennen können, wie viele Wind tage es in der von ihm auf der Karte ausgewählten Stadt im letzten Jahr gab. Dabei hat der Benutzer die Möglichkeit, nicht nur die Stadt, sondern auch die Windgeschwindigkeit auszuwählen, die für die Definition eines Windtages relevant ist.
Das Dashboard bietet eine benutzerfreundliche Oberfläche, die es ermöglicht, detaillierte Informationen über die Häufigkeit von Windtagen in verschiedenen Zeiträumen zu erhalten. Der Benutzer kann durch Auswahl einer bestimmten Stadt und einer spezifischen Windgeschwindigkeit sehen, wie viele Tage pro Monat im letzten Jahr die gewählte Windgeschwindigkeit erreicht oder überschritten wurden. Diese Funktion ist besonders nützlich für die Analyse saisonaler Muster und für das Verständnis der windreichen Zeiten des Jahres.
Darüber hinaus ermöglicht das Dashboard einen Jahresvergleich der Wind tage zwischen zwei verschiedenen Städten bei derselben Windgeschwindigkeit. Dies bedeutet, dass der Benutzer nicht nur die Daten für eine einzelne Stadt sehen kann, sondern auch einen direkten Vergleich zwischen zwei Städten anstellen kann. Diese Vergleichsfunktion ist hilfreich, um Unterschiede im Windverhalten zwischen verschiedenen geographischen Standorten zu erkennen und zu analysieren.

Projektmanagement

Da unser Team neu zusammengestellt wurde und wir uns vorher nicht kannten, entschieden wir uns nach dem ersten Workshop, gemeinsam in eine Bar zu gehen, um uns besser kennenzulernen und auszutauschen. Dadurch konnten wir nicht nur eine persönliche Verbindung aufbauen, sondern auch die Aufgaben entsprechend den Stärken jedes Einzelnen verteilen. Um dies zu erleichtern, erstellten wir eine Skillmatrix, die uns eine visuelle Darstellung unserer Fähigkeiten bot.
Benjamin, der als Informatiker arbeitet, brachte das grösste Know-how in Bezug auf Programmierung mit. Silas, der durch seine Berufserfahrung bestens mit Projektarbeit vertraut ist, übernahm das Projektmanagement. Adem, der über gute Fähigkeiten in der Datensuche und -beschreibung verfügt, war für die Erstellung der Datenstory, sowie der Datenquellen suche verantwortlich. Trotz dieser Rollenverteilung war es uns wichtig, voneinander zu profitieren und nicht isoliert an unseren jeweiligen Aufgaben zu arbeiten. Wir wollten die Challenge gemeinsam als Team bestreiten und nicht als eine zufällig zusammengewürfelte Gruppe.
Wir hielten wöchentliche Meetings ab, in denen wir die Aufgabenverteilung regelmässig überprüften und neu festlegten, Probleme und Erfolge diskutierten sowie die Checklisten der Workshops durchgingen. Diese Meetings ermöglichten es uns, voneinander zu lernen und gemeinsame Entscheidungen zu treffen, um unseren Zeitplan einzuhalten.
Unser Zeitplan war eng an die Checklisten gebunden, was uns eine strukturierte Herangehensweise ermöglichte und das Vertrauen gab, die Challenge pünktlich abzuschliessen. Die regelmässigen Meetings und die klaren Verantwortlichkeiten sorgten dafür, dass jeder im Team wusste, was zu tun war, und wir als Einheit effektiv zusammenarbeiten konnten.

Technisches

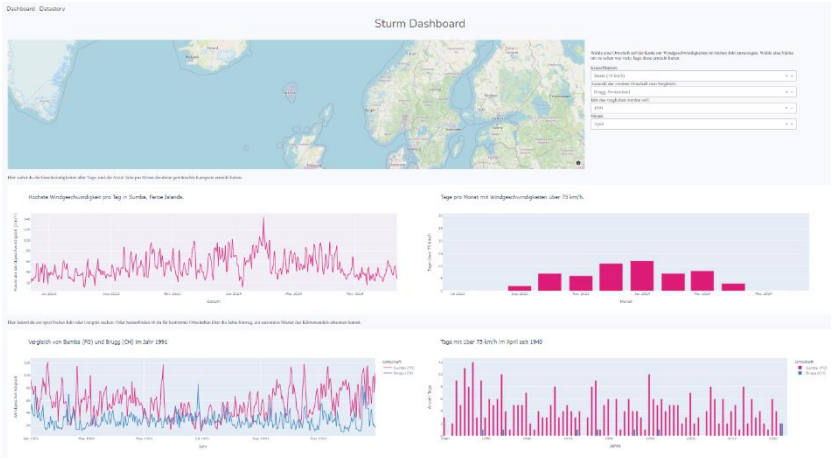
Framework

Wir haben uns entschieden, Django als Framework für unsere Webseite zu verwenden, und das aus mehreren Gründen:

1. **Schnelle Entwicklung**: Django bietet eine Vielzahl integrierter Tools und Features, die die Entwicklung beschleunigen. Mit seinem «all-inclusive»-Ansatz bringt es viele vorgefertigte Komponenten mit, die uns geholfen haben, schnell eine funktionsfähige Webseite zu erstellen.
2. **Klares und sauberes Design**: Django fördert ein sauberes und pragmatisches Design. Durch das Prinzip „Don't Repeat Yourself" (DRY) wird Redundanz vermieden, was den Code effizienter und wartbarer macht. Dies half uns, eine gut strukturierte und leicht verständliche Codebasis zu entwickeln.
3. **Skalierbarkeit**: Django ist sehr skalierbar und eignet sich sowohl für kleine als auch für grosse Projekte. Es kann problemlos mit dem Wachstum unserer Webseite und der Anzahl der Benutzer umgehen. Dies würde uns die Flexibilität geben, unser Projekt in der Zukunft problemlos zu erweitern.
4. **Community und Dokumentation**: Django verfügt über eine sehr aktive Community und umfangreiche Dokumentation. Dies bedeutet, dass wir bei Problemen oder Fragen schnell Hilfe finden konnten. Die gut dokumentierten Anleitungen und Tutorials haben uns geholfen, effizient zu lernen und unsere Ziele zu erreichen.
5. **Integriertes Admin-Interface**: Django bietet ein integriertes Administrations-Interface, das uns die Verwaltung unserer Datenmodelle erheblich erleichtert hat. Mit minimalem Aufwand konnten wir ein leistungsfähiges Backend entwickeln, das uns erlaubte, Inhalte einfach zu verwalten und zu aktualisieren.
6. **ORM (Object-Relational Mapping)**: Django kommt mit einem leistungsstarken ORM, welches die Datenbankabfragen stark vereinfachte. Dies ermöglichte uns, mit Datenbankoperationen auf einer höheren Abstraktionsebene zu arbeiten, was die Entwicklung beschleunigte und den Code lesbarer machte.

Oberfläche

Nach dem Aufrufen unserer Webseite findet der Benutzer oben im Menü die Option, direkt unsere Datenstory anzusehen, oder er bleibt auf dem Dashboard, wo ihn als erstes eine farbige Europakarte, ein Dropdown-Menü sowie zwei Graphen erwarten.



Mithilfe dieser interaktiven Karte, die ähnlich wie bei Google Maps zoom- und verschiebbar ist, kann der Benutzer eine Wetterstation auswählen. Die Windgeschwindigkeiten der gewählten Stadt werden dann im Graphen links unter der Karte für das vergangene Jahr als Liniendiagramm angezeigt. Daneben zeigt ein weiterer Graph, wie häufig es im letzten Jahr pro Monat Windgeschwindigkeiten gab, die der Benutzer im Dropdown neben der Karte gewählt hat.

Über das Dropdown-Menü kann der Benutzer zusätzlich eine zweite Stadt sowie ein Jahr und einen Monat auswählen, um diese mit der zuvor gewählten Stadt aus der Karte im Abschnitt „Langzeitvergleich und Saisonalität“ zu vergleichen. Um einen spezifischen Monat genauer zu betrachten, kann der Benutzer einen bestimmten Bereich im Graphen durch Markieren vergrössern. Diese Funktion steht in allen Graphen zur Verfügung.

Die Graphen wurden mithilfe der Python-Bibliothek Plotly (plotly, 2024) erstellt. Plotly bietet zahlreiche Vorteile, die für uns relevant waren:

- Interaktive Visualisierungen ermöglichen es Benutzern, mit den Daten zu interagieren, Filter anzuwenden, Details anzuzeigen und die Darstellung dynamisch anzupassen, was besonders wichtig ist, um komplexe Datenmuster zu erkennen und zu präsentieren.
- Plotly bietet eine umfangreiche Palette von Diagrammtypen und Funktionen zur Visualisierung, darunter Balken-, Linien- und Scatterplots, Heatmaps, 3D-Diagramme und mehr. Diese Vielfalt erlaubte es uns, Daten auf verschiedene Arten darzustellen und die passende Visualisierung für unsere spezifischen Analyse- und Präsentationsbedürfnisse auszuwählen.
- Die nahtlose Integration mit Python war für unser Team besonders vorteilhaft, da wir Python als unser Hauptentwicklungswerkzeug verwenden.

Insgesamt unterstützt uns Plotly dabei, hochwertige, interaktive Datenvisualisierungen zu erstellen, die nicht nur informativ sind, sondern auch die Benutzererfahrung auf unserer Webseite deutlich verbessern.

Datenbank

Wie bereits im Kapitel zur Datenauswahl erläutert, haben wir eine API genutzt, um Wetterdaten abzurufen. Dadurch entfiel die Notwendigkeit, die umfangreichen Wetterdaten lokal zu speichern. Statt dessen haben wir uns entschieden, die Standorte der Wetterstationen in unserer PostgreSQL-Datenbank zu speichern. Dies war erforderlich, da die API die Koordinaten der Wetterstationen benötigt, um die entsprechenden Wetterinformationen bereitzustellen.
Durch das Speichern dieser Standortdaten in unserer Datenbank stellen wir sicher, dass Benutzer nicht selbst nach den Koordinaten ihrer Vergleichsstadt suchen müssen. Statt dessen können sie einfach den Namen ihrer gewünschten Stadt eingeben, und unser Programm übernimmt den Rest der Koordinatenermittlung automatisch für sie. Dies vereinfacht die Benutzererfahrung erheblich und optimiert den Prozess der Datensuche und -abfrage über unsere API-basierte Plattform.

Literaturverzeichnis

DeepL. (21. 06 2024). *DeepL Write*. Von https://www.deepl.com/de/write?utm_source=lingueede&utm_medium=linguee&utm_content=homepage_write abgerufen

open-Meteo. (21. 06 2024). *Historical Weather API*. Von https://open-meteo.com/en/docs/historical-weather-api abgerufen

plotly. (21. 06 2024). *plotly*. Von https://plotly.com/ abgerufen

Für dieses Dokument wurde DeepL Write als Textkorrektur verwendet. (DeepL, 2024)