Visualizing Historical and Current Weather Trends

Danil Eramasov, Nurmukhammet Adagamov, Bulat Fakhrutdinov

February 16, 2025

1 Project Goal and Vision

The goal of this project is to develop an interactive web application that visualizes historical and current weather trends. The primary focus is on comparing temperature variations over time to provide insights into climate change and weather patterns.

Key Features and Purpose

- **Temperature Change Comparison Map**: Users can select two years (e.g., 2010 and 2020) to compare temperature differences across regions.
- **Interactive Heatmaps**: Display historical temperature data using color-coded visualizations.
- **Time-Series Graphs**: Show temperature variations over time for selected locations.
- **Filtering & Tooltips**: Users can filter by country, region, or time range, with tooltips providing exact values.

Target Audience

- Researchers and students analyzing climate trends.
- Journalists reporting on climate change.
- General users interested in understanding how weather has changed in their area.

2 Dataset Description

Data Source and Collection Plan

We will collect weather data from the following sources:

- **NOAA API** (link): Historical temperature records and climate data.
- **Meteostat API** (link): Historical and real-time weather data.
- **Web Scraping (Backup Option)**: If API limitations exist, we will scrape historical weather data from sites like Weather Underground, World Weather Archive 1, World Weather Archive 2 using Scrapy.

Data Content

- **Variables**: Temperature, humidity, wind speed, precipitation.
- **Time Period**: 1930 - Present (if available).
- **Geographic Coverage**: Global, with zoom functionality for specific regions.

3 Visualization App Architecture

Pipeline Overview

The project follows a standard data visualization pipeline:

1. **Data Collection**: We will use *requests* and *httpx* to fetch data from weather APIs like OpenWeatherMap and NOAA, and *Scrapy* for scraping historical weather data from relevant sources if needed. The collected data will be stored in JSON format for easier processing.

2. **Data Cleaning & Preprocessing**: Using Pandas and NumPy, we will clean and preprocess the data by handling missing values, normalizing date formats, and structuring it for effective visualization. Data will be aggregated by region, time periods, and weather attributes.

3. **Data Exploration & Processing**: Before visualization, we will conduct exploratory data analysis (EDA) using Jupyter Notebook, Matplotlib, and Seaborn to identify patterns, trends, and anomalies in weather changes over different years. This will help define meaningful insights for visualization.

4. **Data Delivery**: A Flask RESTful API will serve the processed weather data in JSON format. The API will provide endpoints to access weather trends by year, location, and comparison between different years. This ensures efficient interaction between the backend and frontend.

5. **Data Visualization**: The frontend will use *D3.js*, *Leaflet.js*, and *Chart.js* to build engaging and interactive visualizations. Planned visualizations include:

   - An interactive heatmap to compare temperature differences between years, where regions are colored based on warming or cooling trends.
   - A time-series graph to visualize temperature and precipitation trends over decades.
   - A map with tooltip pop-ups displaying detailed temperature data for specific years and locations.

4 Proposed Visualizations and Features

- **Choropleth Map**: Displays temperature differences using a color-coded world map.
- **Time-Series Graph**: Historical temperature trends for a selected location.
- **Interactive Heatmap**: Represents weather changes over time.
- **Filtering & Tooltips**: Users can select specific locations, years, or weather variables.
- **Responsive UI**: A clean, mobile-friendly interface using Bootstrap.

5 Project Timeline and Milestones

Week	Task
Week 1	Finalize dataset sources, API setup, define project scope.
Week 2	Implement data scraping/APIs and preprocessing pipeline.
Week 3	Develop Flask API and start exploratory data analysis (EDA).
Week 4	Begin frontend development (UI, map integration).
Week 5	Implement key visualizations (maps, charts, filtering).
Week 6	Add interactivity and refine UI/UX.
Week 7	Final testing, debugging, and deploy web app.
Week 8	Prepare project presentation and submit final version.

6 Resources for UI Templates

Since we want a professional design without spending excessive time on CSS, we will use:

- **Bootstrap Themes**: themes.getbootstrap.com
- **AdminLTE (Dashboard UI)**: adminlte.io
- **Tailwind UI**: tailwindui.com

7 Final Tech Stack

Component	**Technology**
Backend API	Flask
Data Collection	NOAA API, Meteostat API, Scrapy (if needed)
Data Processing	Pandas, NumPy
Frontend Visualization	D3.js, Leaflet.js, Chart.js
UI Framework	Bootstrap, HTML/CSS