Overview

Bird Sighting Data Visualization is an interactive web application that represents the Spacial - Temporal visualization of bird sightings. This enables the researchers to explore and analyse the various patterns in bird species sightings over time, seasons across the geographic locations. The raw data that was collected as a part of bird sighting observation was cleaned and converted to feed the data visualization technology so that clear patterns both spatially and temporally can be achieved for the research purpose.

Dataset Overview

- BirdData.csv and SpeciesCode.csv are the two data sets that were provided for us. **BirdData.csv is an observation dataset** and **SpeciesCode.csv is a metadata**.
- BirdData.csv included columns like Location ID, Latitude, Longitude, Species Code, Day, Month, Year, Data Entry Technique, Counts and so on. This gave information about the event-based, time-stamped and measured data.
- SpeciesCode.csv gave the **lookup information** to help interpret the observational data like common name, scientific name along with the Species Code. This doesn't contain any measurements.
- BirdData.csv has 100000 records and SpeciesCode.csv has 15966 species of birds.

Technology Stack and Tools:

Frontend: Used for building the interactive bird sightings visualization

- **HTML5** – Structure of the web page.
- **CSS3** – Styling.
- **JavaScript** – Dynamic interaction and data binding.
- **Leaflet.js** – Interactive maps for geospatial plotting.
- **EChart.js** – Visual chart representation of bird sightings over time.
- **Flatpickr** – Used for picking the dates for Bird Sightings.
- **Chroma.js** – Used for marking each species with a different color.

Backend: Used for data cleaning, merging, transformation, and exporting

- **Python (3.x)** – Core scripting language for data wrangling.
- **Pandas** – Tabular data manipulation and CSV/JSON handling.
- **NumPy** – Numerical operations and data handling.
- **Jupyter Notebook** – Interactive data analysis environment.

Development Tools:

- **Visual Studio Code (VS Code)** – Code editor for web development.
- **Jupyter Notebook** – Data cleaning and exploration.
- **Linux Terminal / Bash** – File and environment management.
- **Git Bash** – Local SSH and SCP operations for file transfer.

Data Formats

- **CSV** – For intermediate cleaned data, analysis, and archive
- **JSON** – For use in frontend JavaScript visualization

Implementation Details

Data Preprocessing: Raw datasets are often messy, incomplete, or inconsistent. Dataset cleaning and transformation are important to remove the noise, fix the inconsistency, transform the data for usability create derived fields for easier filtering and visualization. Reduced data size and removal of irrelevant column improves the performance.

- **The Dataset was cleaned and transformed** using **Python (pandas, numpy libraries)** via Jupyter.
- Cleaning involved identifying the null values and analysing the impact of their removal and decided to **remove the null value** records. Removed certain columns like sub_id, proj_period_id and obs_id that were not relevant to this particular data visualization.
- Cleaning also involved setting the latitude and longitude values within the boundaries. Since **latitude** must be between **-90 and 90 degrees**, and **longitude** between **-180 and 180 degrees**, any rows containing values outside these ranges were flagged as invalid.
- Transforming the dataset includes the **merging of BirdaData and SpeciesCode** datasets. This enriched the raw observational dataset with meaningful species information.
- Merged three columns (Day, Month and Year) into a Date column and used the month column to compute the season. Improve the **readability and interpretability** of the dataset for users and researchers.
- Standardized the column names by converting them to lowercase, stripping the spaces and so on.
- Converted to .csv easy to read, write, and version-control. JSON format to use the cleaned data directly in the **frontend web visualization** (JavaScript).

Visualization:

- **Map View (Spatial Visualization)** - Displays bird sightings across a geographic map using species-specific markers. On page load, all species are visualized across the map. On selecting the particular bird species in the filter you can see the distribution of the birds across the region (North America as per the data). This map helps in the analysis of bird species distribution in East coast, West Coast and other parts of the continent based on season and snow depth as well.

  Users can click on each marker to reveal a **tooltip** displaying:
  - Common and scientific name of the bird
  - Date of the sighting
  - Season
  - Location (lat/lon)
  - Bird count

- **Line Chart (Species Count Over Time) -** Plots a temporal trend of sightings for the selected bird species. The X-axis represents months, while the Y-axis shows the count of sightings. Useful for **analyzing seasonal patterns and migration trends**.

- **Bar Chart (Top Sighted Species) -** Highlights the most frequently sighted species in the dataset. Helps identify **dominant species** based on the filtered parameters. Specially if we have selected all species filters and based on the changes of the season we can see the top-sighted bird species for the particular season.

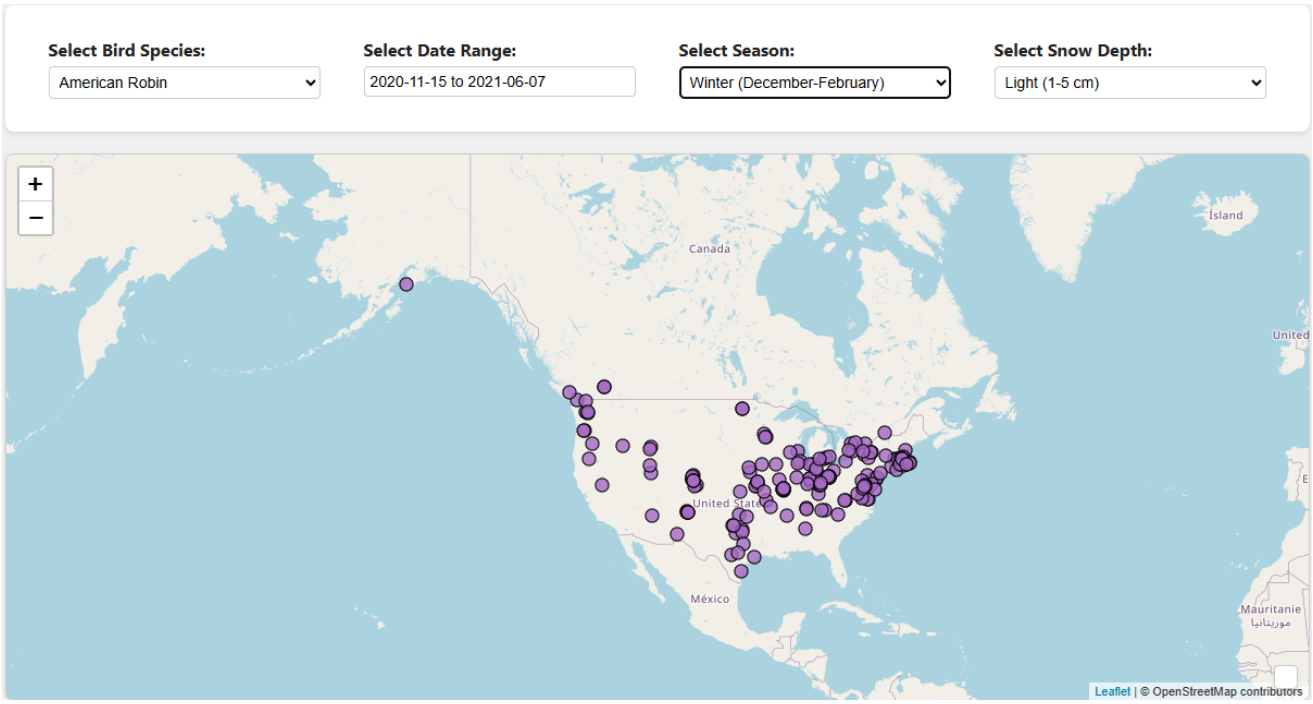
- **Interactive Filters** - The visualization includes multiple filters for tailored exploration. These filters dynamically update the map and charts, enabling users to conduct a **focused, meaningful exploration** of the dataset.
    - **Bird Species Selector**: Choose a particular species to focus the map and charts.
    - **Date Range Filter**: Filter sightings by specific start and end dates.
    - **Seasonal Filter**: Quickly group sightings by Winter, Spring, Summer, or Fall.
    - **Snow Depth**: Explore how snow conditions might impact sightings.

- **Downloadable Data (Dataset)** - A **cleaned and transformed dataset (CSV)** is also made available for download. This allows researchers and data analysts to
    - Reproduce the visualizations.
    - Conduct further offline analysis.
    - Validate the data sources used in the visualization.

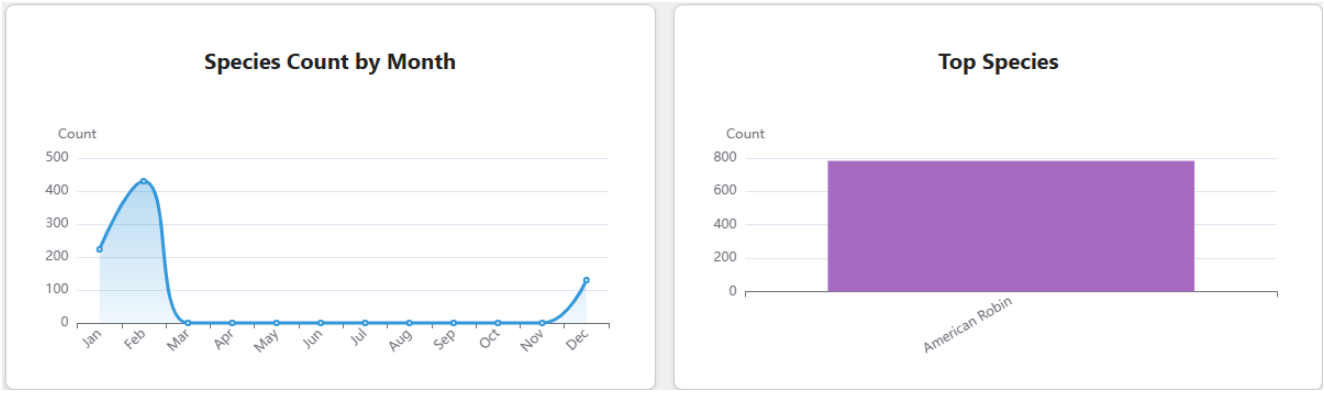
- **About Section** - An “About” section is included in the web interface to explain the purpose and scope of the visualization. Describe the source and cleaning of the data. Provide usage instructions for users unfamiliar with the tool.

- **Report:** A detailed overview of the Bird species visualization dashboard.

- **Example Usage Scenario** - A user interested in American Robin can filter by “American Robin,” select a specific date range (e.g., 15th Nov 2020 to 7th June 2021), select snow depth “Light (1-5 cm)” and choose “Winter” as the season. The map will update to show only American Robin sightings in that range. Hovering over markers reveals detailed sighting data, while the line chart shows monthly sighting trends.



Pic: A dynamic map showing bird sightings with species specific markers.



Pic: Charts that show trends over time and frequency of sightings

Web App Design: This application is organized into clear folders and supports dynamic user interactions. The app follows these core principles:

- **Simplicity**: Minimalist layout with intuitive interactions
- **Responsiveness**: Usable on various screen sizes.
- **Modularity**: Clear separation of HTML5, CSS3, JavaScript for maintainability
- **Transparency**: Data source and transformation steps openly shared in the "About" section

The application files are organized under the **WebApp** directory inside the main project folder BirdVisualization_akan291.

BirdVisualization_akan291/

```
├── WebApp/
    ├── index.html          # Main HTML5 entry point
    ├── styles.css          # CSS3 styles for layout and design
    ├── scripts.js          # JavaScript logic for interactivity and rendering
    ├── Cleaned_BirdData_15June.json  # Cleaned dataset used in visualization
```

Other supporting project folders:

- **scripts**: Python scripts for preprocessing and utilities
- **notebooks**: Jupyter notebooks for data cleaning and merging
- **Data**: Raw and processed CSV/JSON datasets
- **output**: Exported images or data artifacts

Challenges: Throughout the development of the Bird Sighting Data Visualization, several challenges were encountered, particularly in setup, environment management, and web deployment. These provided valuable learning opportunities and contributed to a deeper understanding of working in remote and virtualized environments.

- **Library installation via traditionally venv instead of uv**:
    - Initially, we created a Python virtual environment using the traditional venv approach, which required installing system-level packages like python3-venv. This introduced unnecessary complexity and elevated privileges (sudo access), slowing down the setup process.
    - After completing the environment setup, we discovered that **modern tools like UV or PIPX** could have streamlined this by creating isolated environments with faster dependency management and no need for venv.

- **Port Tunnelling for Jupyter and Web Server Access:**
    - It was not initially obvious how to expose the web server or Jupyter Notebook running on the VM to the local browser. Default attempts using **localhost:8000** failed, leading to confusion about the correct tunnelling method.
    - Implemented SSH port forwarding to bridge the remote and local environments: **ssh -i /path/to/private-key.pem -L 9000:localhost:8080 username@hostname**
    - This allowed us to serve the visualization via **http://localhost:9000/**

- **Transferring files between the local and virtual machine:**

    - Transferring JSON, HTML, CSS, and JS files from the local development environment (VS Code) to the remote VM was initially challenging due to SSH permission errors and incorrect file paths.
    - Used **scp** (Secure Copy Protocol) with the correct path and key to push files to the VM. Correct file permissions and absolute paths are crucial. Having a clear folder structure made this process smoother after the first successful transfer.

- **Cleaning and merging datasets:**
    - The original datasets had missing values, redundant columns, and invalid geographic coordinates outside of acceptable latitude/longitude ranges.
    - Dropped rows with missing essential values. Merged BirdData.csv with SpeciesCode.csv using a left join on species_code. Computed and added a new **season** column for temporal filtering. Removed invalid coordinates to ensure map compatibility.

Deployment: Remote deployment on virtual machine with port tunnelling

Since the project is hosted on a remote VM, and direct access to the web server is not exposed publicly, **SSH port tunneling** was used to access the app securely.

- **Prerequisites:**
  Private SSH Key
  Remote access granted to the VM (Username and Hostname/IP)
- **SSH into the remote VM:**
  ssh -i /path/to/private-key.pem username@hostname
- **Navigate to the notebooks in Juptyer Labs, then click on the New button and then click on terminal and then further navigate to the WebApp directory:**
  cd ~/Desktop/BirdVisualization_akan291/WebApp
- **Start a local Python HTTP server on the VM:**
  python3 -m http.server 8080
- **In a separate terminal on your local machine**, **establish an SSH tunnel**:
  ssh -i /path/to/private-key.pem -L 9000:localhost:8080 username@hostname
- **Open your browser and access the visualization at:**
  http://localhost:9000/index.html

Challenges and Learnings

- Developed a complete end-to-end data visualization pipeline from raw bird sightings data to a fully functional, browser-based interactive dashboard.
- Cleaned and merged datasets (BirdData.csv and SpeciesCode.csv) using Python and Jupyter Notebook, ensuring accuracy and readiness for visualization.
- Built an interactive web application using HTML5, CSS3, JavaScript, and libraries like Leaflet.js,EChart.js,Flatpickr,Chroma.js to allow users to:
    - Explore bird sightings spatially (map) and temporally (charts).
    - Apply filters by species, date range, season, and snow depth.
    - Download the cleaned dataset directly from the web interface.

- Gained hands-on experience in:
    - Remote environment management via SSH, SCP, and port tunnelling.
    - Environment isolation with Python venv.
    - Writing modular and responsive frontend code without any frameworks.

Data preparation is crucial: Clean, structured, and well-joined data enables accurate and meaningful visualizations.

Efficient communication between backend data prep and frontend design is key to building usable tools.

Modern tooling (like uv) can greatly simplify Python setup, which was a learning in hindsight.

Interactive design using plain JavaScript can still result in a powerful and responsive application when done thoughtfully.

Understood how to work in a **VM-based assessment environment** using terminal tools, remote access, and Jupyter effectively.

Enhancements:

- **Enable the search for the filters**: This allows the user to type the bird species name and easy to filter the data.
- **Add Summary Panel:** Show average sightings, most active months, or rare species sightings. Add environmental context to the map like elevation or temperature overlays.
- **Backend Integration**: While we used a static JSON file for the front end, this architecture could be enhanced using Node.js or Flask to serve the data via APIs. Since Node.js and npm were preinstalled, the platform supports such an extension if real-time querying or data security were required.
- **Predictive Sightings with ML (using prophet lib):** This project could incorporate machine learning models to predict bird sightings based on historical data and environmental conditions. Using models such as time-series prediction/forecasting we could estimate the likely presence of specific bird species in given regions and times. This would help researchers not only visualize past observations but also anticipate future trends in bird behavior, making the dashboard a proactive tool for ecological planning and research.