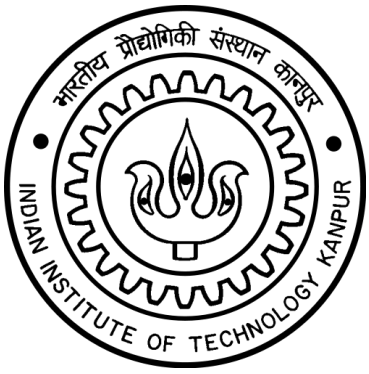

Project Report For CS661: BIG DATA VISUAL ANALYTICS

2024-2025 Summer Semester

American Wildfire Data Visualization

Group Number - 11

Dornipati Sai Suhruth - 230391 (ssuhruth23@iitk.ac.in)
Srijani Gadupudi - 231033 (srijanig23@iitk.ac.in)
Manemoni Pavan Kumar - 230630 (manemonip23@iitk.ac.in)
Aravapalli Govind Sujith - 230182 (agsujith23@iitk.ac.in)
Chilamakuri Kundan Sai - 230330 (ckundans23@iitk.ac.in)
Derangula Sujith - 230351 (sujithd23@iitk.ac.in)
Macha Mohana Harika - 230612 (mharika23@iitk.ac.in)
Rudraksh Sukhdev Kawde - 220921 (rudraksh22@iitk.ac.in)

Contents

Chapter 1

Introduction

1.1 Project Overview and Goals

Wildfires are a significant environmental and societal challenge with far-reaching consequences.Understanding their patterns, trends, and underlying causes is critical for effective prevention, resource management, and policy-making. The United States, with its vast and diverse ecological regions, faces increasing wildfire risks due to factors such as climate change, prolonged droughts, and human activity. While government agencies and research institutions collect extensive wildfire data, accessing, integrating, and visualizing this information in a meaningful and actionable way remains a complex task.

The **"American Wildfire Data Visualization"** project was undertaken to address this challenge. The primary goal is to develop a user-friendly, interactive web application. The dashboard aims to:

- Provide a centralized platform for accessing wildfire data statistics from 1992-2015 all across united states of america.
- Enable comprehensive exploration of wildfire data by geography (state-wise), time (yearly and seasonal trends), and underlying causes.
- Facilitate comparative analysis across regions, time periods, and wildfire causes to identify patterns and anomalies.
- Employ data visualization techniques to reveal patterns, hotspots, and correlations that might be obscured in raw data tables.
- Incorporate basic machine learning techniques to uncover deeper insights and potential future trends.

1.2 Scope

The scope of this project encompasses:

- **Data Integration:** Loading, cleaning, and standardizing the US Wildfires dataset (1992–2015) from Kaggle, which contains 1.88 million records with geospatial, temporal, and organizational attributes.
- **Visualization Development:** Creating interactive visualizations such as geospatial maps (state-wise, county-wise), line charts, bar charts, heatmaps, and predictive analysis outputs.
- **Web Application Interface:** Building a multi-tabbed, interactive web dashboard to host the visualizations and provide user-friendly controls for filtering and exploration.
- **Interactivity:** Implementing dynamic filtering options by geography (state-level), time (yearly and seasonal trends), and underlying causes to enable comprehensive exploration of wildfire data.
- **Machine Learning Features:** Developing a classification model to predict wildfire causes based on historical patterns and fire attributes.

1.3 Technologies Used

The dashboard is built using a modern full-stack architecture and leverages the following technologies:

- **FastAPI:** A high-performance web framework for building RESTful APIs in Python. Used for designing the backend services and handling API requests efficiently.
- **PostgreSQL:** A robust relational database system with support for geospatial data. Used for storing, querying, and processing wildfire records and geographic data.
- **React.js:** A JavaScript library for building component-based user interfaces. Used to develop the frontend of the web application, enabling a dynamic and responsive dashboard.
- **Recharts.js:** A popular open-source charting library built specifically for React.js. It allows developers to quickly and easily build interactive and customized charts using a declarative, component-based approach.
- **Deck.gl:** A WebGL-powered framework for visualizing large-scale geospatial data. Used for rendering wildfire incidents on interactive maps with high performance.

- **Plotly.js:** A JavaScript graphing library for creating interactive, publication-quality visualizations (line charts, bar charts, heatmaps, etc.) in the frontend.
- **D3.js:** A powerful JavaScript library for producing dynamic, interactive data visualizations in web browsers. Used for implementing custom visual elements and enhancing user interaction in maps and charts.
- **Docker:** A containerization platform used for packaging the application and its dependencies into containers, ensuring consistency across development and deployment environments.
- **Machine Learning Libraries:**
  - **Scikit-learn:** Used for developing the classification model to predict wildfire causes.
  - **LightGBM:** A gradient boosting framework that uses tree-based learning algorithms. Applied for building efficient, high-performance predictive models for wildfire cause analysis.
  - **Pandas:** Utilized for data cleaning, preprocessing, and manipulation of large wildfire datasets.
  - **Numpy:** Employed for numerical computations and array operations during data processing.
  - **imbalanced-learn:** This library is specifically designed to help with classification problems where the distribution of classes is imbalanced.
- **GeoJSON:** Standard format for encoding a variety of geographic data structures, used to load and overlay geographic boundaries on maps.

1.4 Tasks Addressed

The dashboard is designed to enable comprehensive exploration and in-depth analysis of wildfire data across multiple dimensions. It equips users with powerful tools to observe trends, compare patterns, and gain predictive insights into wildfire behavior. The key tasks include:

- **Organizational Geospatial Analysis:** Visualize wildfire incidents across different regions and jurisdictions, allowing users to compare spatial patterns and understand how land ownership and responding agencies influence wildfire dynamics.
- **Temporal Pattern Analysis:** Explore seasonal, annual, and diurnal trends in wildfire occurrences using interactive line charts and heatmaps, helping identify high-risk periods and recurring patterns over time.

- **Comparative Performance Metrics:** Assess the effectiveness of various agencies in containing wildfires by analyzing key metrics such as fire size, duration, and containment times, providing a basis for performance benchmarking.
- **Major Incident Analysis:** Investigate large fire complexes by visualizing how multiple fires merge into major incidents and examining their composition, spread, and management over time.
- **Predictive Cause Analysis:** Leverage integrated machine learning models to predict the likely causes of wildfires based on historical data and fire attributes, supporting proactive decision-making and risk mitigation.
- **Trend Observation and Hotspot Identification:** Utilize heatmaps to identify geographic and temporal hotspots of wildfire activity, revealing concentration zones and enabling strategic resource allocation.
- **Cause-Based Analysis:** Analyze and compare various wildfire causes to understand their prevalence, regional distribution, and impact, offering insights into underlying factors and potential preventive measures.

Chapter 2

Data Engineering and Preprocessing

A visual analytics dashboard is only as reliable as the data that powers it. The foundational data for this project originated from a raw SQLite database file. While this file contained a wealth of information—over 1.88 million fire records—it was not structured for the fast and complex queries demanded by an interactive, real-time application. Therefore, a significant data engineering process was essential to transform this raw data into a clean, structured, and high-performance PostgreSQL database ready for analysis.

This chapter provides a detailed account of the custom data pipeline, encapsulated in the Python script `run_data_pipeline.py`, which was built to perform this critical transformation.

2.1 Data Sources

The project integrates data from three distinct sources to provide a comprehensive view of wildfire incidents. Two sources provide the core fire and agency data, while a third provides the geographic data necessary for map-based visualizations.

1. **The `Fires` Table:** This is the primary dataset, sourced from the `FPA_FOD_20170508.sqlite` file. It contains 1.88 million individual fire incident records from 1992 to 2015, forming the transactional backbone of our analysis.

2. **The `NWCG_UnitIDActive_20170109` Table:** A crucial lookup table provided within the same SQLite file. It serves as a dimensional table, translating cryptic agency codes from the `Fires` table (e.g., `CAHNU`) into full, human-readable names and other organizational details. This enrichment is vital for adding meaningful context to the fire records.

3. **Geographic Boundary Data:** To render the county-level heatmap on the dashboard's frontend, a separate GeoJSON file named `geojson-counties-fips.json` was utilized. This file contains the geometric map polygons for all counties in the United States. Each county shape is identified by a unique Federal Information

Processing Standard (FIPS) code, which acts as a foreign key to link our processed fire data directly to the map for visualization.

2.2 The Data Processing Pipeline: An Overview

To bridge the gap from the raw SQLite file to the final analytical database, a script was developed using Python, leveraging the Pandas library for in-memory data manipulation and the SQLAlchemy library for robust database communication.

The primary architectural challenge was the sheer volume of the `Fires` table. Attempting to load all 1.88 million rows into memory at once would be highly inefficient and likely exceed the available system resources. To circumvent this, the pipeline was engineered to process the data in a **streaming fashion**—reading, cleaning, and loading the data in smaller, manageable batches or "chunks." This design choice ensures that the entire process runs smoothly with a low and constant memory footprint, making the pipeline scalable and robust.

The script is also designed to be idempotent, meaning it can be run multiple times with the same outcome. Each time it is executed, it rebuilds the database table from scratch, ensuring the data is always fresh and perfectly consistent with the latest version of the processing logic.

2.3 Detailed Pipeline Stages

The data processing pipeline is broken down into several distinct stages, each with a specific purpose. These stages are executed sequentially for each chunk of data that is processed, ensuring a methodical and error-resistant workflow.

2.3.1 Stage 1: Creating a Clean Slate

Before any new data is introduced, the pipeline first prepares the PostgreSQL database to ensure a pristine environment.

- **Action:** The script connects to the database and executes a command to completely delete the existing `wildfires` table if it exists. It then immediately recreates a new, empty `wildfires` table based on the precise structure defined in the project's SQLAlchemy models.
- **Reasoning:** This initial step is fundamental to the pipeline's reliability. It makes the process idempotent, meaning it produces the same result no matter how many times it is run. This prevents the accumulation of duplicate records from multiple runs and guarantees that the database table perfectly mirrors the data structure that our application's backend expects.

2.3.2 Stage 2: Extracting and Combining Data

With a clean database table ready, the script begins extracting data from the source SQLite file.

- **Action:** The script reads the large `Fires` table not in one go, but in manageable chunks of 50,000 rows. For each of these chunks, it performs a merge operation, joining it with the much smaller `NWCG_UnitIDActive_20170109` lookup table which has been pre-loaded into memory.
- **Reasoning:** Reading the data in chunks is the core strategy for maintaining memory efficiency. The merge operation enriches the raw, code-heavy fire records with descriptive, human-readable information like the full agency name. A "left" merge was specifically chosen to ensure that no fire records are accidentally dropped during this process, even if a corresponding agency code is missing or invalid in the lookup table.

2.3.3 Stage 3: Cleaning and Transforming the Data

This is the most intensive stage of the pipeline, where the raw data within each chunk is rigorously cleaned and reshaped. The most significant challenge in this stage was rectifying the inconsistent and error-prone date and time information. The original temporal columns suffered from numerous quality issues, including missing values, non-standard text entries, and incorrect numeric formats. A careful, multi-step process was engineered to construct reliable and standardized datetime values.

1. **Standardizing Time Strings:** A custom Python function was implemented to parse and standardize the time columns. It was designed to:
   - Handle missing values by defaulting to midnight (`'0000'`)
   - Pad incomplete numbers to four-digit length (e.g., `930` $\rightarrow$ `0930`)
   - Insert a colon to produce standard `HH:MM` format
2. **Converting Julian Dates:** The discovery and containment dates, originally stored in Julian day format, were converted to standard calendar dates. This robust process:
   - Gracefully handled errors and invalid entries
   - Converted problematic values to `NaT` (Not a Time)
3. **Combining Date and Time:** The cleaned time strings and converted date objects were combined to create complete datetime values for:
   - Fire discovery `(DISCOVERY_DATETIME)`

- Fire containment (`CONT_DATETIME`)

This operation used safe functions to ensure invalid components resulted in invalid datetimes.

2.3.4 Stage 4: Creating New Features (Feature Engineering)

Once the data was clean—especially the critical date and time values—it became possible to derive new, insightful columns that did not exist in the original dataset. This process, known as "feature engineering," adds significant analytical value by creating variables that can directly answer specific questions about wildfire patterns.

- **Action:** Using the newly reliable `DISCOVERY_DATETIME` and `CONT_DATETIME` columns, the script calculates several new data columns for each fire record.
- **New Features Created:**
  - `FIRE_DURATION_DAYS`: Calculated by finding the difference between the containment time and the discovery time. This feature is fundamental for analyzing the efficiency of firefighting responses and the overall scale of a fire incident.
  - `DISCOVERY_MONTH`: The month in which the fire was discovered. This is essential for analyzing the seasonal patterns of wildfires, such as identifying a distinct "fire season."
  - `DISCOVERY_DAY_OF_WEEK`: The day of the week the fire was discovered. This can help identify whether human-caused fires are more prevalent on weekends versus weekdays, pointing to recreational activities as a potential factor.
  - `DISCOVERY_HOUR`: The hour of the day the fire was discovered. This provides insight into diurnal (daily) patterns, such as whether fires are more likely to be discovered during daylight hours.

2.3.5 Stage 5: Loading the Data into the Database

This is the final stage of the pipeline, where the fully cleaned, enriched, and feature-engineered chunk of data is loaded into the PostgreSQL database.

- **Action:** Before loading, the script performs one last, critical cleaning step. It replaces all special Pandas values for missing data (`NaN` for numbers and `NaT` for datetimes) with Python's standard `None` value. The clean data is then efficiently inserted as a batch into the `wildfires` table.
- **Reasoning:** This final replacement is a crucial step for database compatibility. Relational databases like PostgreSQL understand Python's `None` as a `NULL` (or

empty) value, but they do not recognize the special missing value objects from the Pandas library. Performing this conversion prevents errors during the data loading process and ensures data integrity.

Once a chunk is successfully loaded into the database, it is cleared from the system's memory, and the pipeline loops back to Stage 2 to process the next 50,000-row chunk. This entire process repeats until all 1.88 million fire records have been systematically processed and loaded. The final result is a clean, robust, and analysis-ready dataset that serves as the foundational data source for the entire visualization dashboard.

Chapter 3

Visualizations of Wildfire Data

In order to translate raw wildfire records into actionable insights, we developed a two-pronged suite of visualizations: temporal charts that reveal patterns over hours, days, and containment durations, and geospatial maps that expose regional hotspots and incident-level correlations. In total, our dashboard ingests over 1.8 million fire records (1992–2015) and renders eight distinct views. The following sections describe each plot in depth—its design rationale, technical implementation, and the key insights it delivers.

3.1 Temporal Analysis Suite

Our temporal suite comprises four complementary charts. All four are powered by the same base dataset (1.8 million records), filtered by user-selected year(s), state(s), and cause(s) via a unified React/MUI filter panel. We preprocess discovery timestamps into discrete hourly bins, weekday labels, and containment-duration intervals; then we pass the cleaned arrays to Plotly.js to render each view with custom layouts, hover-templates, and performance optimizations (e.g. `useResizeHandler`, `style={ width: '100%', height: '100%' }`).

3.1.1 Top 12 Fire Causes (Radial Log Bar)

To visualize the categorical breakdown of wildfire causes, we implemented a D3-powered radial bar chart. Each bar represents one of the top twelve ignition causes by frequency, such as Debris Burning, Lightning, or Arson, and all other less frequent causes are grouped under an "Other" category. The bars are arranged angularly around a circle, with their radial lengths log-scaled to accommodate the wide variation in counts.

This design choice allows both common and rare causes to be represented perceptibly within a single view. Color intensity encodes magnitude: the more fires caused by a category, the deeper its red tone (based on the `interpolateYlOrRd` colormap). A centered numerical label dynamically updates to show either the total count or a hovered category's count. This chart is fully interactive, tooltipped, and responsive across devices.

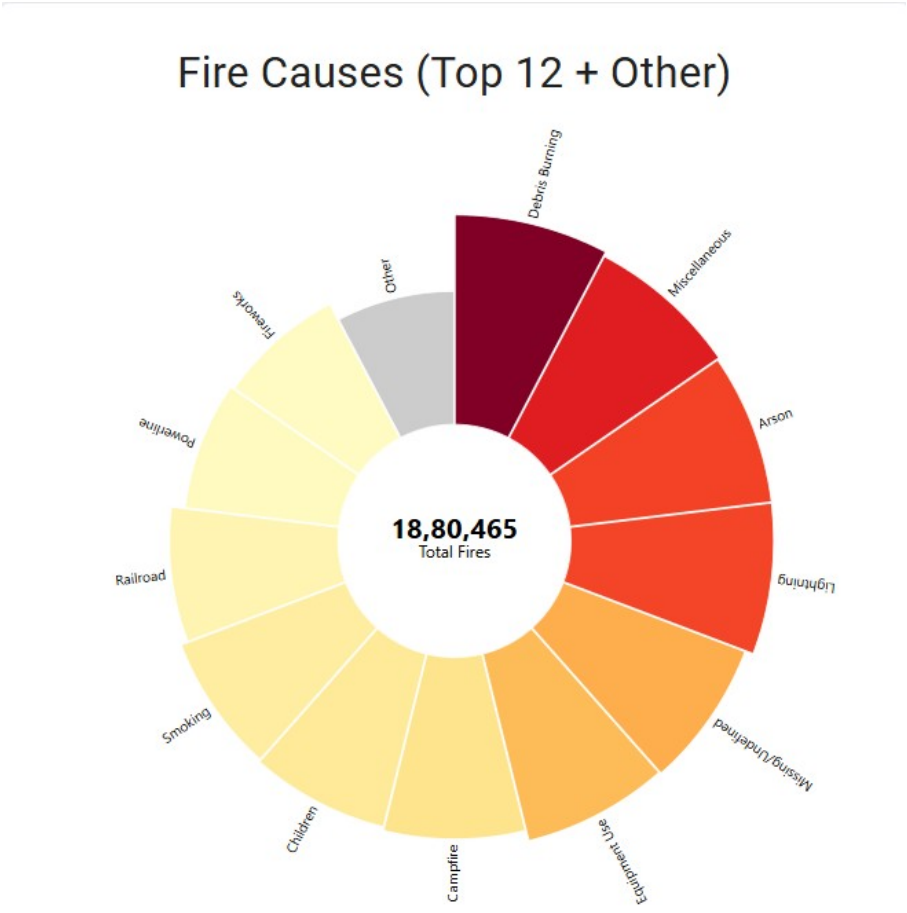

Figure 3.1: Radial bar chart showing the top 12 wildfire causes (plus "Other") using log-scaled bar lengths. Color intensity reflects fire count.

3.1.2 Seasonal Wildfire Trends Over Years

To analyze seasonal dynamics of wildfires across years, we aggregate monthly discovery counts into four canonical seasons: Winter (Dec–Feb), Spring (Mar–May), Summer (Jun–Aug), and Fall (Sep–Nov). The line chart uses year on the x-axis and fire count on the y-axis to plot each season as a separate trend line.

This view reveals clear temporal patterns: Summer exhibits the sharpest and most consistent peaks across states, often due to hot, dry weather and increased human activity. Spring and Fall activity varies more subtly, influenced by regional vegetation cycles and weather systems. Some states show non-trivial Winter activity, often in coastal or southern regions. The chart uses Recharts for clean SVG rendering, tooltips, and legend-based toggling. It updates dynamically with state filters and is suitable for both desktop and mobile.

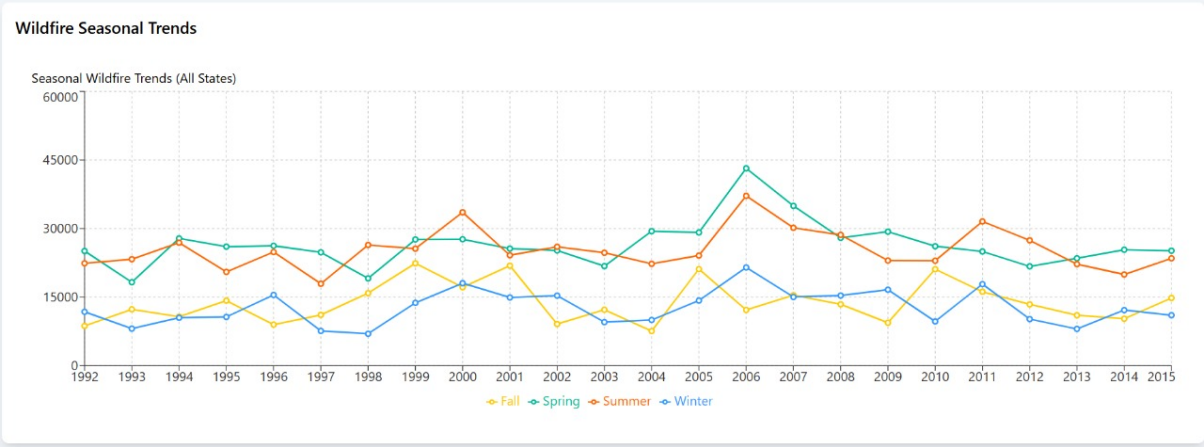

Figure 3.2: Multi-line chart showing seasonal wildfire activity (Winter, Spring, Summer, Fall) across years. Summer peaks dominate consistently.

3.1.3 Diurnal Cycle of Fire Discoveries

To effectively visualize the diurnal rhythm of wildfire ignition, we transform each fire's discovery hour into an angular coordinate for a 24-hour radial chart using Plotly's `barpolar` trace. This method provides a natural representation of cyclical time, enabling users to intuitively grasp patterns across the day. The chart is constructed from over 1.6 million records with valid hourly timestamps, revealing a strong surge in wildfire detection from late morning to early afternoon—approximately 10AM to 3PM local time. This period aligns with peak ambient temperatures, low humidity levels, and elevated human activity, all of which contribute to heightened ignition likelihood. The radial chart uses `theta=hour*15` to map hours to degrees and `r=fire_count` to determine bar lengths, styled with a vivid orange tone (`rgba(255,140,0,0.8)`) reminiscent of fire embers.

To support readability, the polar axis is auto-scaled to the highest bin count (nearing 150,000 fires) and clearly labeled at cardinal clock positions (12AM, 6AM, 12PM, 6PM). Hover templates enable precise inspection of fire counts per hour, offering both interactivity and detail. The chart layout is fully responsive, leveraging Plotly's `useResizeHandler` and flexible styling to adapt seamlessly across screen sizes—from widescreen monitors to tablets. This design ensures maximum accessibility without compromising information density or aesthetics.

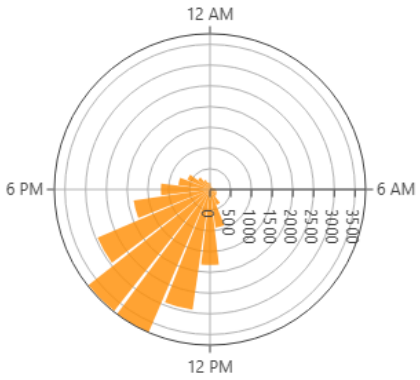

Figure 3.3: Radial bar chart of hourly fire discoveries (0–23 h), showing a clear late-morning/early-afternoon peak in ignition and detection.

3.1.4 Weekly Fire Cadence by Cause

Wildfires driven by human activity often follow a weekly cadence, whereas those ignited by lightning bear no such pattern. To dissect these dynamics, we aggregate and stack daily counts from Monday through Sunday into four primary cause categories: Direct-Human, Indirect-Human, Lightning, and Miscellaneous/Undefined. The resulting stacked-bar chart uses Plotly's `barmode='stack'` and a consistent grey-scale palette for print-friendliness. With average daily totals ranging from about 20,000 to 28,000 fires, the chart reveals that Indirect-Human causes surge by up to 40% on weekends—clearly visible as pronounced weekend humps—while Lightning remains uniform throughout the week. Custom hover-templates show both per-cause and total-day counts, and the x-axis respects the natural Monday–Sunday order via `categoryorder='array'`.

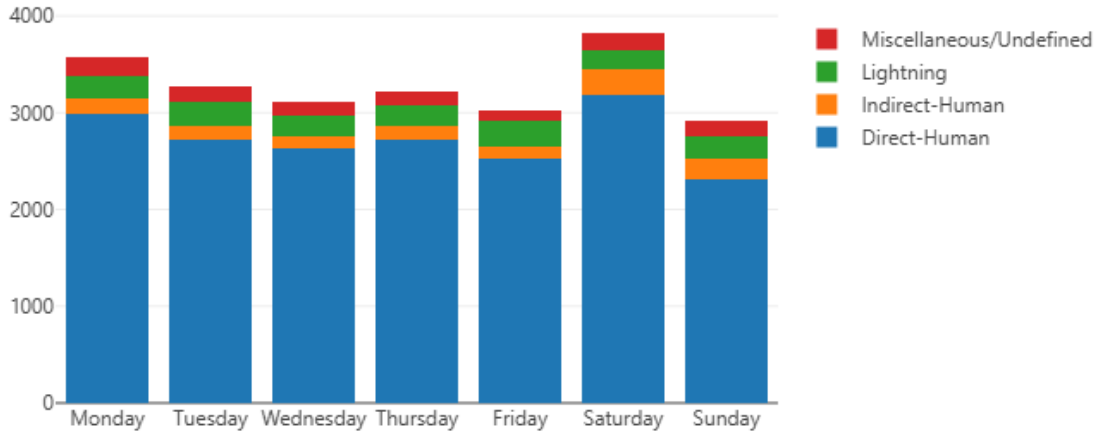

Figure 3.4: Stacked bar chart of daily wildfire counts (Mon–Sun), color-coded by ignition cause, revealing weekend peaks in human-caused fires.

3.1.5 Containment Duration Distribution (Logarithmic Scale)

Containment times exhibit a heavy-tailed distribution: the vast majority of fires are contained within one day, but a small fraction persists for weeks. We bin durations into daily intervals (0–1 day, 1–2 days, . . . , 30+ days) and plot the fire-count per bin using a spline-smoothed area (`mode='lines+markers'`, `fill='tozeroy'`, `line.shape='spline'`) with a log-scaled y-axis (`type='log'`). This approach preserves fine detail in both the dense bulk (0–1 day bin with about 1.6 million records) and the long tail (30+ day bin with roughly 2,300 records). A semi-transparent green fill (`rgba(34,197,94,0.3)`) underscores containment efficacy at low durations while emphasizing the tail risk. The log scale enables planners to identify the long-haul threshold—around five days—beyond which resource demands escalate sharply.

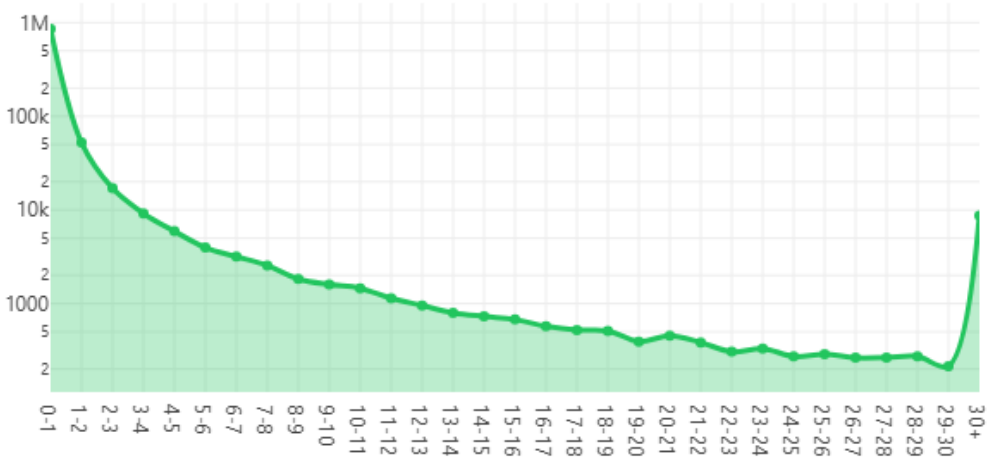

Figure 3.5: Area chart of fire counts by daily containment-duration bins, displayed on a logarithmic y-axis to capture both common and exceptional cases.

3.1.6 Size Class vs. Cause Breakdown

The relationship between fire size and ignition cause yields critical prevention insights. We classify incidents into size classes A (0–1 acre), B (1–10 acres), C (10–100 acres), D (100–1,000 acres), E (1,000–10,000 acres), F (10,000–100,000 acres), and G (100,000+ acres); then we stack-plot counts by cause (Arson, Debris Burning, Lightning, Miscellaneous, Other). Displayed horizontally to accommodate long class labels, this chart shows that Debris Burning dominates Class A (over 60% of about 1.2 million small burns), while Lightning and Other causes drive Classes F and G (each accounting for over 50% of the few hundred largest events). We use a neutral color ramp for print and offer distinct hatch patterns in CSS for screen variants. Users can click legend items to toggle causes on and off, facilitating focused analysis.

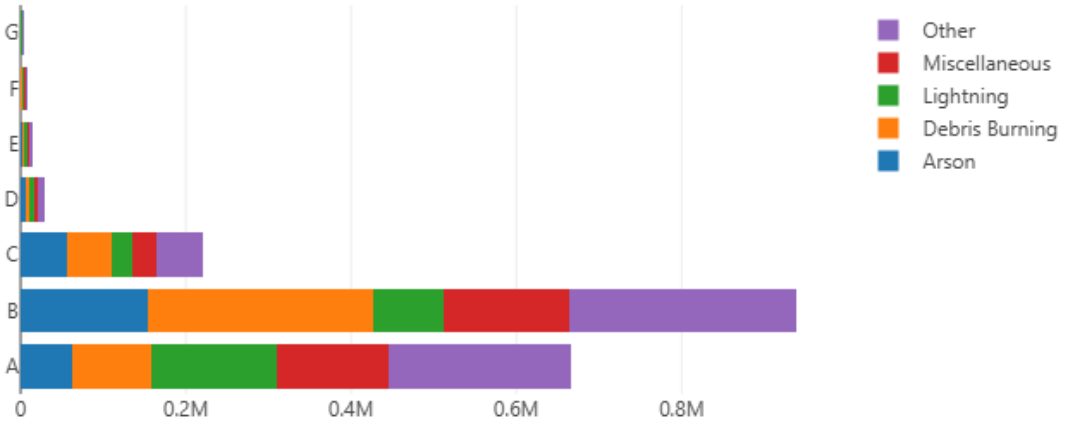

Figure 3.6: Horizontal stacked-bar chart showing wildfire counts across size classes A–G, segmented by ignition cause.

3.2 Geospatial Analysis Suite

To complement our temporal charts, we built an interactive map suite using Deckġl and React-Map-GL. All maps draw from a PostGIS-backed API and respond dynamically to year, date-range, state, and cause filters via React context. We employ raw scatterplots, clustering, and choropleth layers to deliver three distinct spatial perspectives.

3.2.1 Raw Fire Points (Scatterplot Layer)

Every fire incident is rendered as a geo-positioned circle, sized by burned area via a square-root scale (5–50 px) and colored from pale blue ($<$ 10 acres) to deep maroon (5,000 acres). To maintain interactivity, we page or year-filter the 1.8 million records down to 50,000 points at once. Hover tooltips reveal fire name, county, cause, and size. This view exposes true density corridors—such as California's Sierra Nevada, Texas panhandle, and Florida peninsula—while highlighting sparse regions.

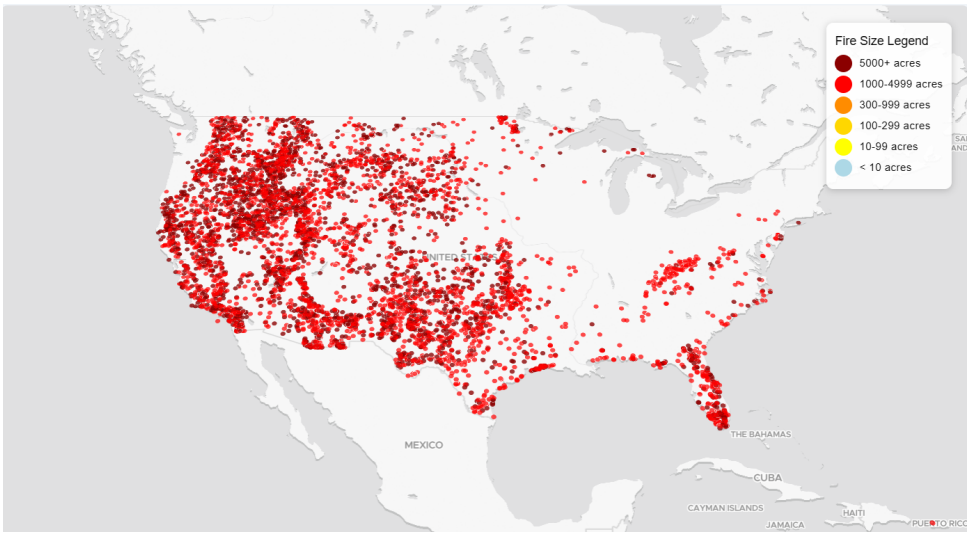

Figure 3.7: Individual fire points plotted by location, with marker size proportional to burned area, revealing geographic density and distribution.

3.2.2 Clustered Fire Points with Labels

At broader zoom levels, clusters group nearby points into green circles labeled by count (e.g. "8,891 fires" in central Kansas). We use `use-supercluster` (75 px radius, maxZoom = 20) to compute clusters on the fly. Clicking a cluster animates the viewport to its expansion zoom, breaking it into smaller clusters or individual points. This approach reduces clutter and communicates regional volumes efficiently.

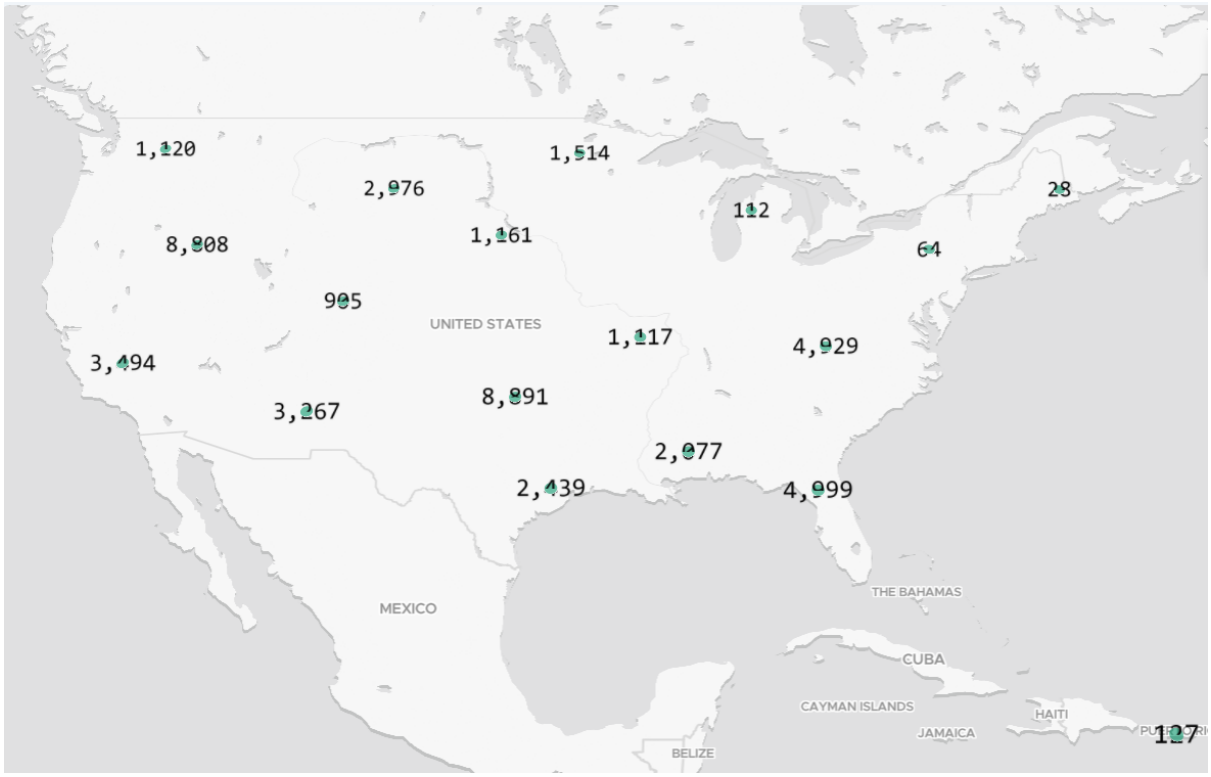

Figure 3.8: Deckġl ScatterplotLayer clusters fire points into labeled groups at national scales, with smooth zoom-driven drill-down.

3.2.3 County-Level Heat Map (Choropleth Layer)

For sub-state granularity, we color each county polygon by total fire count in the current filter window. Counts range from single digits (pale yellow) through mid-hundreds (orange) to thousands (deep red). Data come from `getCountyData`, and colors update via Deckġl's `GeoJsonLayer` with `updateTriggers`. Hover tooltips display county name and count (e.g. Grays Harbor County: 467 fires). This choropleth reveals spatial patterns such as multi-county burn complexes and allows fine-grained seasonal comparisons.

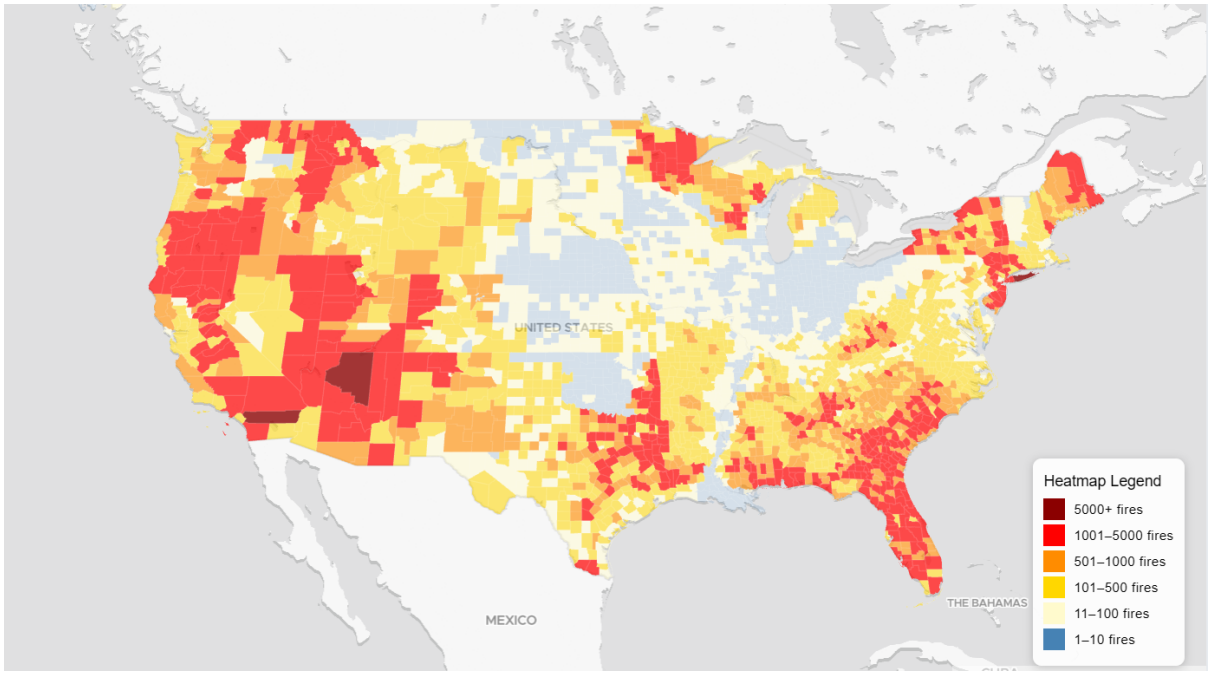

Figure 3.9: Choropleth of county fire counts, dynamically filtered by date or year, highlighting regional hotspots at sub-state scale.

Chapter 4

Prediction Model Pipeline

This Chapter details the machine learning pipeline developed to predict wildfire causes. The process was iterative, involving several key stages designed to overcome the significant challenges posed by the dataset, such as severe class imbalance and the need for more predictive features. We will explain the rationale behind our data preparation steps, the specific feature engineering techniques employed, the selection of an advanced resampling method, and the final model architecture and training process. Each step was crucial in building a robust and insightful classification model.

4.1 Data Preprocessing For Model Training

A critical early decision was to filter out vague and non-actionable target classes, specifically 'Missing/Undefined' and 'Miscellaneous'. While this initially appeared to lower the overall accuracy score, it was a necessary step to force the model to make a definitive choice between known causes, dramatically increasing the practical utility and "honesty" of the final predictions.

4.2 Feature Engineering for Predictive Modeling

To maximize the information available to the model, the following features were engineered:

- **Cyclical Temporal Features:** The `DISCOVERY_DOY` (Day of Year) was transformed into `doy_sin` and `doy_cos` components. This was essential for the model to understand the cyclical nature of seasons, a key factor for causes like lightning.
- **Categorical Encoding:** All categorical text features (e.g., `STATE`, `NWCG_REPORTING_AGENCY`) were converted into numerical integer codes. This was a necessary step to resolve technical errors with the resampling library (`SMOTEENN`) which requires all inputs to be numeric.

4.3 Handling Class Imbalance With SMOTEENN

The dataset is severely imbalanced, where some causes have hundreds of thousands of samples while others have only a few hundred. To address this, the **SMOTEENN** technique was employed. This advanced hybrid method was chosen over simpler techniques for two reasons:

1. **SMOTE (Oversampling):** It first creates new, synthetic examples of the rare fire causes (e.g., 'Structure', 'Powerline'), giving the model more data to learn their patterns.
2. **ENN (Cleaning):** It then cleans the resulting dataset by removing noisy or ambiguous samples that lie on the border between classes. This creates a clearer decision boundary for the model to learn from.

This two-step process results in a higher-quality, balanced training set than oversampling alone.

4.4 Model Selection and Training

A **Light Gradient Boosting Machine (LightGBM)** classifier was selected due to its good performance, speed and efficiency on large, tabular datasets. An aggressive set of hyperparameters was used to train a complex model capable of learning the detailed patterns in the data. The model was trained on the resampled data using early stopping to prevent overfitting.

4.5 Model Performance

Analysis:

- **Overall Performance:** The model achieved a final accuracy of **57%**. However, a more representative metric for this imbalanced dataset is the **weighted average F1-score of 0.60**, which indicates a reasonably strong and balanced predictive performance across all classes.
- **Strengths of the Model:** The model demonstrates excellent performance in identifying **Lightning**-caused fires, achieving a high precision of **0.90** and an F1-score of **0.83**. This suggests that the model successfully learned the strong geographic and seasonal patterns associated with natural ignitions.
- **Effectiveness of Resampling:** The success of the SMOTEENN technique is evident in the recall scores for rare classes. For instance, the model correctly identifies **75%** of all **Fireworks**-caused fires and **55%** of all **Railroad**-caused fires, which is a significant improvement over initial models that largely ignored these categories.

- **Areas for Improvement:** The model struggles with classes that have overlapping features or very few samples, such as **Structure** (F1-score of 0.10) and **Powerline** (F1-score of 0.16). This indicates that the available features (e.g., location, time) are not distinct enough to consistently separate these causes from others.

- **Precision vs. Recall Trade-off:** The results show interesting trade-offs. For **Debris Burning**, the model has high precision (0.71), meaning when it predicts this cause, it is often correct. However, its recall is lower (0.42), meaning it misses many actual debris-burning incidents. Conversely, for **Fireworks**, the recall is high (0.75), but the precision is low (0.34), suggesting the model finds most firework-related fires but also incorrectly labels other types of fires as such.

Table 4.1: Final Classification Report for the Focused Model

Cause	**Precision**	**Recall**	**F1-Score**	**Support**
Arson	0.68	0.52	0.59	27188
Campfire	0.47	0.57	0.52	10349
Children	0.28	0.49	0.35	5406
Debris Burning	0.71	0.42	0.53	33697
Equipment Use	0.34	0.41	0.37	9512
Fireworks	0.34	0.75	0.46	1999
Lightning	0.90	0.78	0.83	43747
Powerline	0.10	0.46	0.16	1509
Railroad	0.17	0.55	0.27	1579
Smoking	0.20	0.34	0.25	4396
Structure	0.06	0.40	0.10	502
Accuracy		0.57		139884
Macro Avg	0.39	0.52	0.40	139884
Weighted Avg	0.67	0.57	0.60	139884

4.6 User-Centric Dashboard Design

The final output of this project is not just a model, but an interactive dashboard designed for analysis and transparency. The user interaction is built around providing satisfactory and honest results:

- **Inputs:** The user provides input via an intuitive interface: location data (State, Latitude and Longitude) is captured by clicking a point on an interactive map, while the Date and Fire Size (in acres) are entered into dedicated input boxes.

- **Outputs & Transparency:** To have a "lie factor" of zero, the dashboard presents results in three key ways:

1. **Top Prediction:** A clear "headline" result showing the single most likely cause.
2. **Probability Distribution Chart:** A radial bar chart displaying the model's confidence for the predicted causes. This is the most critical element for user satisfaction, as it honestly communicates the model's uncertainty.
3. **Global Feature Importance Chart:** A static chart that explains which factors the model weighs most heavily overall, adding a layer of explainability to the entire system.

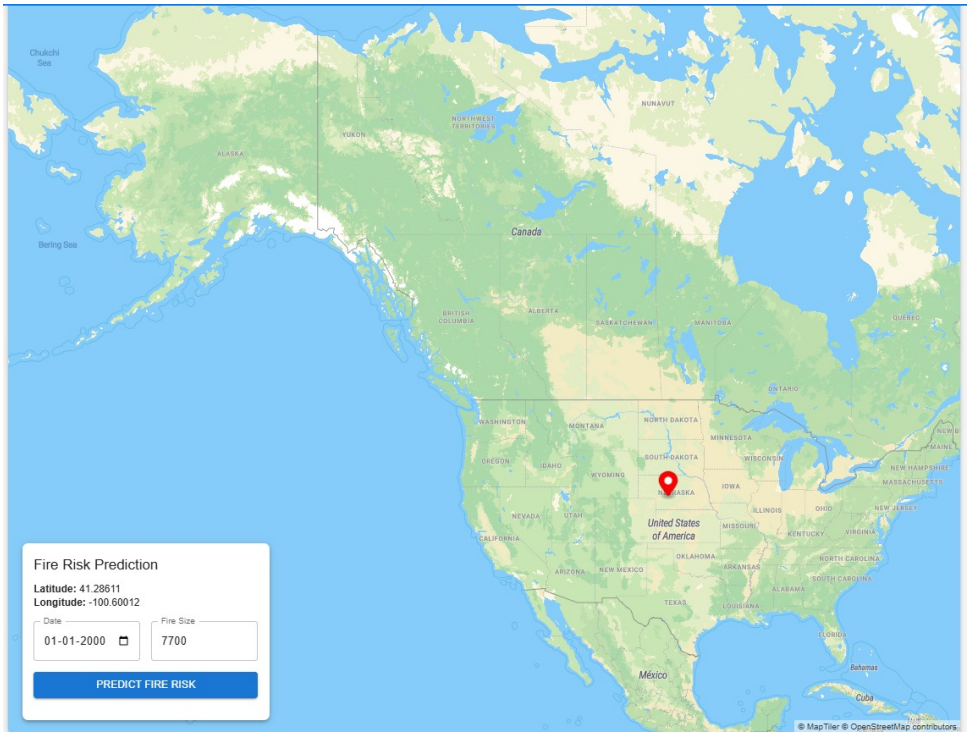

Figure 4.1: Prediction Input Map

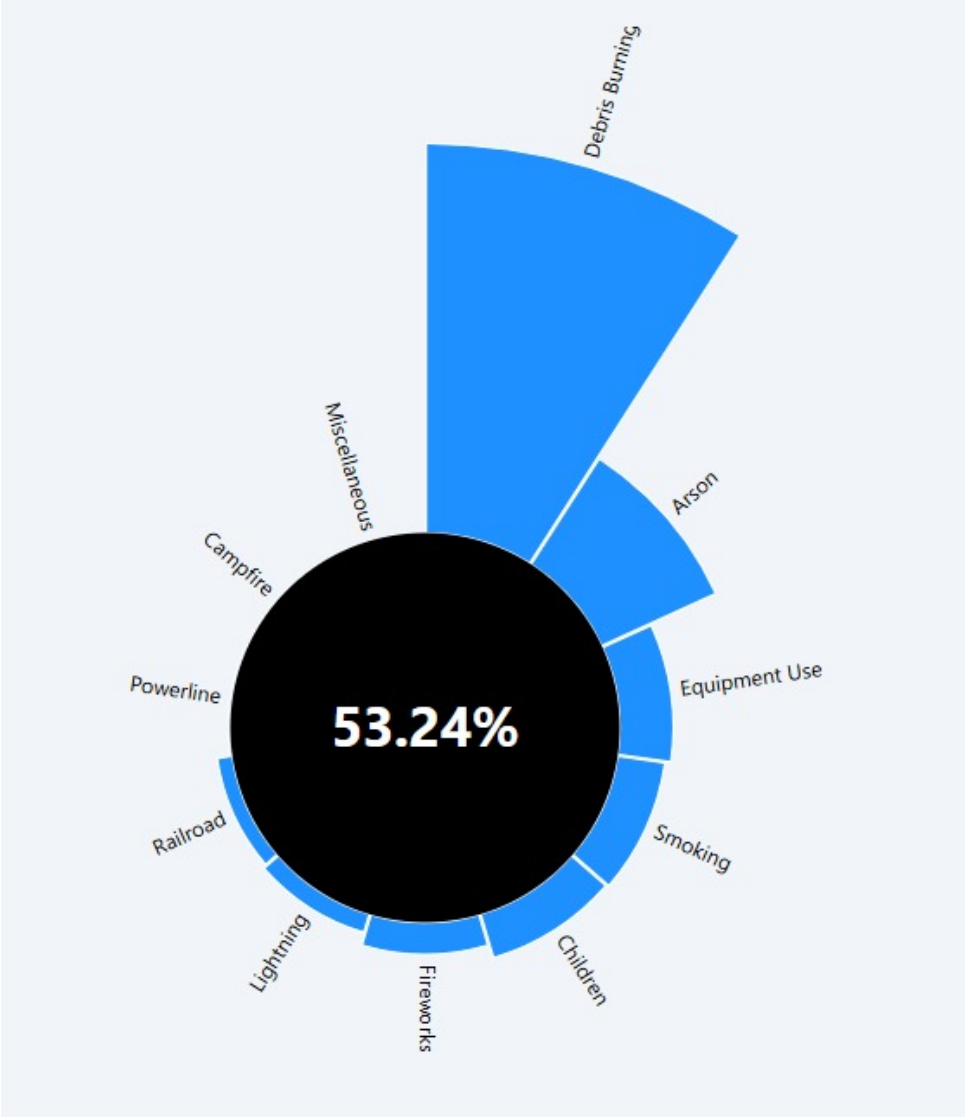

Figure 4.2: Radial Bar Probability Chart

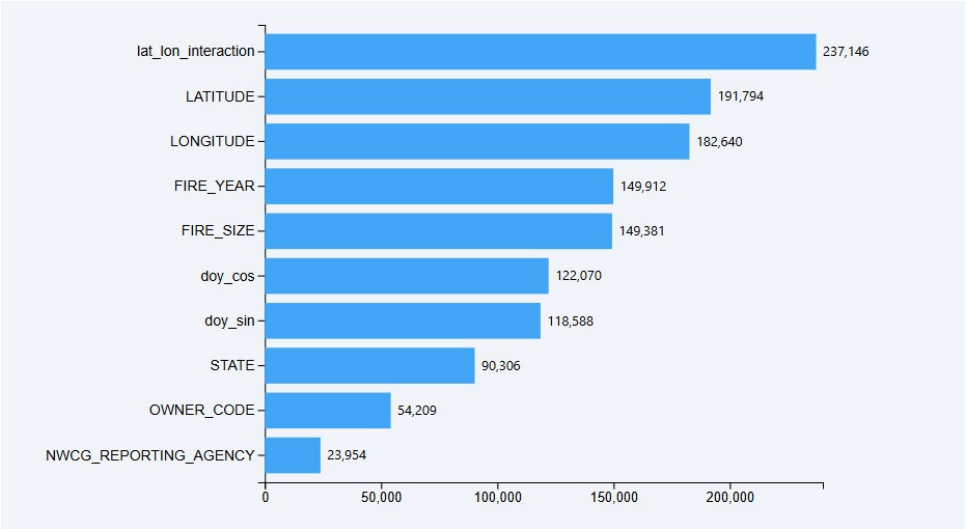

Figure 4.3: Global Feature Importance Chart

Chapter 5

Contributions

Table 5.1: Contributions Made by Each Team Member

Student Name	**Specific Contributions**
A. Govind Sujith	Data cleaning and integration of the US Wildfire dataset (1992–2015); contributed to geospatial visualizations .
Rudraksh Kawde	Worked on frontend UI/UX design.
Srijani Gadupudi	Designed geospatial visualizations and contributed to backend development, implementing some endpoints for geospatial and temporal data queries.
Manemoni Pavan Kumar	Built and deployed a containerized FastAPI backend with analytical endpoints, optimized data ingestion for all records into PostgreSQL, and automated CI/CD with Docker.
Chilamakuri Kundan Sai	Created geospatial visualizations using Deck.gl and contributed to frontend UI/UX design and interactive filtering integration.
Derangula Sujith	Designed and implemented interactive charts (bar, line, and heatmaps) using Plotly.js and D3.js.
Macha Mohana Harika	Frontend UI/UX design, interactive filters, and user experience enhancements in React.js.
Dornipati Sai Suhruth	Integrated machine learning models (LightGBM and Scikit-learn) for predictive cause analysis and backend integration for predictions.

Challenges Resolved:

- Handling the large size (1.88 million records) of the US Wildfire dataset efficiently.
- Merging multiple GeoJSON files to achieve a complete map of the US with state boundaries.
- Reconciling mismatches between geographic names in GeoJSON and dataset using fuzzy matching techniques.

Chapter 6

Challenges and Limitations

While the US Wildfire Data Visualization Dashboard provides powerful tools for exploration and analysis, the project faced several challenges and has some limitations:

- **Data Timeliness:** The dataset covers incidents from 1992 to 2015. As wildfire patterns have evolved due to climate change and urban expansion, insights derived may not fully reflect current conditions.
- **Data Quality and Completeness:** Some records had missing or incomplete attributes (e.g., cause of fire, containment times), requiring imputation or exclusion during analysis.
- **Geo-Spatial Data Alignment:** Ensuring consistency between fire incident coordinates and administrative boundaries was challenging due to variations in GeoJSON datasets.
- **External Data Limitations:** The analysis does not currently incorporate relevant external factors such as weather conditions (temperature, humidity, wind speed) or vegetation types, which are crucial for understanding wildfire behavior. Integrating such data could provide more comprehensive insights but was constrained by availability and compatibility challenges.
- **Computational Constraints:** Rendering large-scale geospatial data (over 1.88 million records) and running machine learning models in real-time required significant computational resources. Limited processing power and memory on the deployment environment imposed constraints on model complexity and visualization responsiveness.
- **ML Model Limitations:** While LightGBM and Scikit-learn models provide useful cause predictions, their accuracy is inherently tied to historical data quality and may not account for emerging or climate-driven wildfire causes.

Chapter 7

Future Work

The current dashboard provides a strong foundation for wildfire data exploration. Future enhancements could include:

- **Data Updates:** Incorporating recent wildfire data (post-2015) from the US Forest Service and other sources to reflect ongoing trends.
- **Expanded Datasets:** Integrating additional datasets such as vegetation maps, climate data (temperature, rainfall), and population density to enable multi-factorial analysis of wildfire risk.
- **Advanced ML Models:** Experimenting with more sophisticated models like XGBoost, SARIMA, or deep learning for cause prediction and time-series forecasting of wildfire incidents.
- **Finer Granularity:** Extending analysis to county or city levels (if data allows) for more localized insights, including dynamic zoom levels in geospatial visualizations.
- **Mobile Support:** Developing a responsive version of the dashboard for mobile and tablet users to broaden accessibility.

Chapter 8

Conclusion

The US Wildfire Data Visualization successfully integrates a large and complex dataset of wildfire incidents spanning 1992–2015 into a single, interactive platform. Built with modern technologies like React.js, FastAPI, Deck.gl, and Plotly.js, the visualization enables users to explore wildfire patterns across geographic, temporal, and organizational dimensions.

Through features such as interactive choropleth maps, heatmaps for hotspot detection, and machine learning-based cause prediction, the system provides a powerful tool for data-driven exploration and decision support. Users can seamlessly transition from national overviews to state-specific and agency-specific analyses, uncovering valuable insights into wildfire behavior and management.

While limitations exist due to data timeliness and computational constraints, the dashboard lays a robust foundation for future enhancements. By transforming raw data into visual narratives and predictive insights, the application contributes to a better understanding of wildfire dynamics and supports proactive wildfire management strategies in the United States.

Code and Deployment Links

The GitHub repository containing the full source code of our project is available at:

- **GitHub Repository:** `GIT REPO`

Note: The deployed dashboard showcases a limited set of functionalities due to the constraints of free hosting services. Features that rely on heavy geospatial data, such as interactive maps rendered with GeoJSON layers, may not function as intended on the hosted version. For full functionality, users are encouraged to run the project locally using the provided setup instructions in the repository's `README.md`.