Final Report

Dhruv Dole   Nathan Rethwisch   Colin Russell
Thanh Mai

Table of contents

1 Introduction/Project Description

Wildfires are a major problem in the United States, costing between $394 billion and $893 billion per year, according to the Joint Economic Committee (U.S. Congress Joint Economic Committee 2023). Identifying where and why wildfires occur is key to preventing wildfires, which protects lives, preserves property, and minimizes economic losses. The goal of this project is to estimate areas of high fire risk based on historical weather patterns and wildfire data. We aim to provide a platform where users can obtain information about past weather conditions and their association with wildfires at a quick glance. This dashboard is designed for a broad audience, including high-level policymakers involved in wildfire prevention strategies and local officials responsible for establishing rules and regulations based on regional wildfire risk.

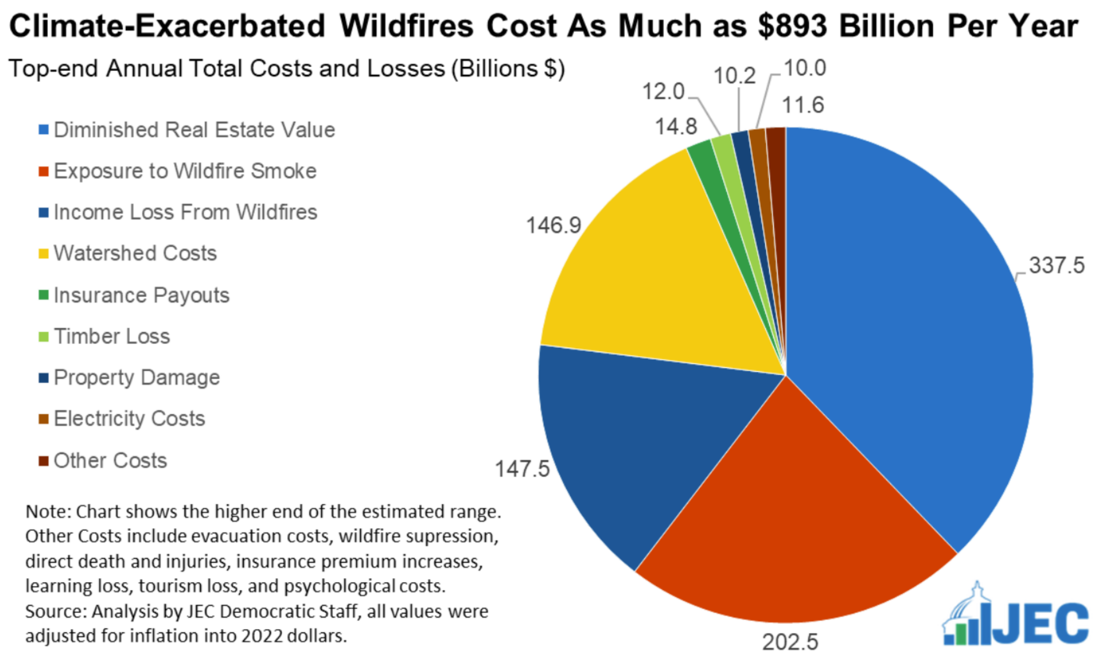


Figure 1: Estimated Cost of Wildfires in 2022

2 Data

```
- Description of sources (api, data format)
  - GHCN daily (temperature, wind, precipitation, snowfall): Issues with lack of docum
  - USFS fire perimeter, fire occurrence (coordinates and perimeters of each fire inci
  - map polygons and rasters
  - other datasets explored but not used: openmeteo api, copernicus data, gnatsgo soil
- Acquisitions, rate limits…
- Process itself
```

2.1 Sources

2.1.1 Global Historical Climatology Network daily (GHCND)

The Global Historical Climatology Network daily (GHCND) is an integrated database of daily climate summaries from land surface stations worldwide. It includes records from over 100,000 stations in 180 countries, providing data on variables such as temperature, precipitation, snowfall, and snow depth. This data is provided in its raw form by the National Oceanic and Atmospheric Administration here. The same data is available in different formats published as an AWS Open Data dataset. The latter dataset was selected because it provides data as comma-separated-values instead of fixed-width tables.

There are two raw 'tables' available from the dataset: `stations` and `daily`. `stations` is available as a single fixed-width text file, and provides a mapping of weather station IDs and geographic coordinates. `daily` is made up of csv records, where each record corresponds to a single observation made by a station. The data is partitioned by year, with a single compressed csv for each year. Source data exists for the year range of 1776 to 2025, however before the year 2000 there is a comparatively small amount of data. It was decided to focus on data from 2000 onwards.

2.1.2 Fire Data

2.2 Acquisition

2.2.1 GHCND

In order to create cleaned tables, it was necessary first to mirror the raw data locally to avoid the need for repeated downloads of the same files. A python script was written which parses the s3 bucket manifest and downloads all `daily` files with the `csv.gz` suffix for years after 1999. The script also downloads a copy of the file `stations.txt`, the raw source for `stations`.

2.2.1.1 First 5 lines of daily file 2010.csv.gz

```
AE000041196,20100101,TMAX,259,,,S,
AE000041196,20100101,TMIN,120,,,S,
AE000041196,20100101,TAVG,181,H,,S,
AEM00041194,20100101,TMAX,250,,,S,
AEM00041194,20100101,TMIN,168,,,S,
```

2.2.1.2 First 5 lines of stations.txt

```
ACW00011604  17.1167  -61.7833   10.1    ST JOHNS COOLIDGE FLD
ACW00011647  17.1333  -61.7833   19.2    ST JOHNS
AE000041196  25.3330   55.5170   34.0    SHARJAH INTER. AIRP            GSN     41196
AEM00041194  25.2550   55.3640   10.4    DUBAI INTL                             41194
AEM00041217  24.4330   54.6510   26.8    ABU DHABI INTL                         41217
```

2.3

2.3.1 Fire Data

2.4 Transformations

2.4.1 GHCND

The transformations for the GHCN Daily data is a single script because the required transformations were changing almost every day, and it was often necessary to rebuild the dataset outright. This also allows for anyone to reproduce the dataset.

The raw daily data includes one column designating the type of observation, and one the actual value recorded. The desired end result was a single table, with a column for each desired observation type or `element`. Once joined with station locations, the data was indexed using the h3 library. This has many benefits which are described below

The basic steps are as follows: `stations.txt` needs to be converted from a fixed-width text file to a typed parquet file. Each daily file needs to be decompressed, typed, and reoriented into the correct format.

The transformation steps are shown in this diagram:

```
graph TD
    A[stations.txt] --> D
    D[read fwf] --> H

    C[raw daily] --> B[decompress]
    B --> E[process dates]
    E --> F[drop unwanted columns]
    F --> G[create single element tables]
    G --> I[join element tables]
    I --> J[join with stations]
    J --> K[generate point geometries]
    K --> L[index with h3]
    L --> H

    H[save as parquet]

    M[read stations.parquet] --> J
```

Once the data was in a viewable format, columns to drop were chosen based by generating percentage nulls. Each time it was decided to keep or drop a column, the column was added or removed from a `keep_columns` list, and the pipeline was re-run. See Table 1 for the resultant table.

Table 1: First 5 rows of 2010.parquet

station_id	year	month	day	prcp	snow	snwd	tmax	tmin	awnd	awdr	evap	latitude	longitude	elevation	state	geometry	Hexagon_ID
CA001011500	2010	1	1	245	0	0	85	35	null	null	null	48.9333	-123.75	75	BC	0x01[illegible]ADD8	[illegible]
CA001011500	2010	1	2	2	0	0	100	55	null	null	null	48.9333	-123.75	75	BC	0x01[illegible]ADD8	[illegible]
CA001011500	2010	1	3	78	0	0	70	30	null	null	null	48.9333	-123.75	75	BC	0x01[illegible]ADD8	[illegible]
CA001011500	2010	1	4	205	0	0	65	50	null	null	null	48.9333	-123.75	75	BC	0x01[illegible]ADD8	[illegible]
CA001011500	2010	1	5	6	0	0	80	50	null	null	null	48.9333	-123.75	75	BC	0x01[illegible]ADD8	[illegible]

2.4.2 Fire Data

- (TODO: go back to code to get exact steps)
- Each transformation applied
- Purpose and issues for each major transformation

2.5 Storage and Query Engine

To handle selections, projections, and group by aggregations on larger-than-memory datasets, PyArrow and Parquet were chosen for their support of lazy evaluation and predicate pushdown, which reduced query time by as much as 700%. These tools also support easy conversion to Pandas dataframes and there is also an Arrow for R package.

The largest operational issue was that the dataset needed to be identical across each developer's machines. To avoid paying for cloud resources, the data was stored in a shared OneDrive folder and symlinked into the local repository by each user, ensuring uniform access paths while allowing individual developers to manage their data access. This setup allowed each user to download the dataset to their local system on demand, while keeping their copy up-to-date.

Initially, the assumption was that the entire dataset needed to be available to the dashboard, but later it was realized that only the 3-day averages produced as part of the model training were necessary. This allowed the dashboard's live data to be loaded into memory on initialization. This means that the dashboard does not require high througput IO access.

3 Exploration and Analysis

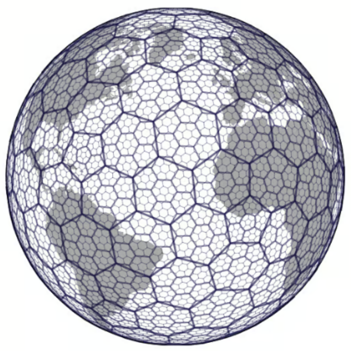


Figure 2: An example of Uber's H3 Cell Grid

After cleaning the data and performing exploratory data analysis, we realized we needed a way to process data that contained both geospatial and time information. One additional challenge we faced was the lack of negative data. Our dataset contained a list of places where fires occurred, but not where they *did not* occur. Thus, this makes it difficult to create a binary prediction model of fire probability, because we only have cases where the fires occurred to train models on. To solve these problems, we turned to Uber's H3 cell grid, shown in Figure 2. H3 cells are a series of hexagonal cells that can be used to cover the surface of the globe at different resolutions. Although perhaps not as precise as certain geographical boundaries for prediction, H3 hexagons are advantageous because of their computational speed and equidistant between neighboring cells. In our case, we fit a grid of cells to the US, with each hexagon representing approximately 4785 square miles, totaling 712 cells. For each day, we marked whether or not a fire occurred in each hex. Meteorological stations within a cell were aggregated to compute mean calculations for weather data. The additional advantage of these cells was that we no longer needed to spatially join the fire and weather data to perform calculations. Once we had the index of the cell, we knew exactly which stations were paired with which fires, saving us the time and computation of continuous joins.

Another challenge was missing data. If all of the weather stations in a cell gave null values for a variable on a certain date, they were imputed with the average from neighboring cells. We also removed precipitation from our analysis because over 50% of the precipitation data was missing.

4 Modeling

4.1 Data Preparation

With the addition of H3 cells and negative data for cells without a fire, we fit a model to predict Y_ij, the probability of a fire occurring in cell i on day j, bounded by $[0, 1]$. We averaged the weather variables over three days, the two days before the fire and the day the fire occurred, for our predictors.
Prediction variables were as follows:

- x_1: Hexagon location
- x_2: Average Elevation
- x_3: Temperature Maximum (3-day average)
- x_4: Temperature Minimum (3-day average)
- x_5: Daily Average Wind Speed (3-day average)
- x_6: Precipitation (3-day average)
- x_7: Snowfall (3-day average)
- x_7: Month

4.2 Model Selection

We split the data into a training and a testing set. The training data ranges from January 1, 2000, to December 31, 2019. The testing data was from January 1, 2020, to March 23, 2025. I encoded the hexagonal location as numeric variables and ran 4 separate models: logistic regression, random forest, XGBoost, and Naive Bayes. After running the models, I then simulated thresholds from 0-1 at a 0.01 interval and selected the optimal prediction threshold for each model that maximized the F1 score, which can be defined as $\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$.

- Precision is defined as how often a positive wildfire prediction is correctly positive. It can be defined as $\frac{\text{TP}}{\text{TP+FP}}$.

- Recall describes the rate at which the model can correctly identify true positives from all possible values. It can be defined as $\frac{\text{TP}}{\text{TP+FN}}$.

F1 score is preferential to accuracy in this case, because the dataset is imbalanced and we have very few actual positive cases of wildfires, so models that maximize accuracy are likely to underestimate the probability of a fire occurring, a more costly mistake than a false positive.

Figure 3 shows the optimal thresholds curves, while Figure 4 displays statistics for each model in the selection process.

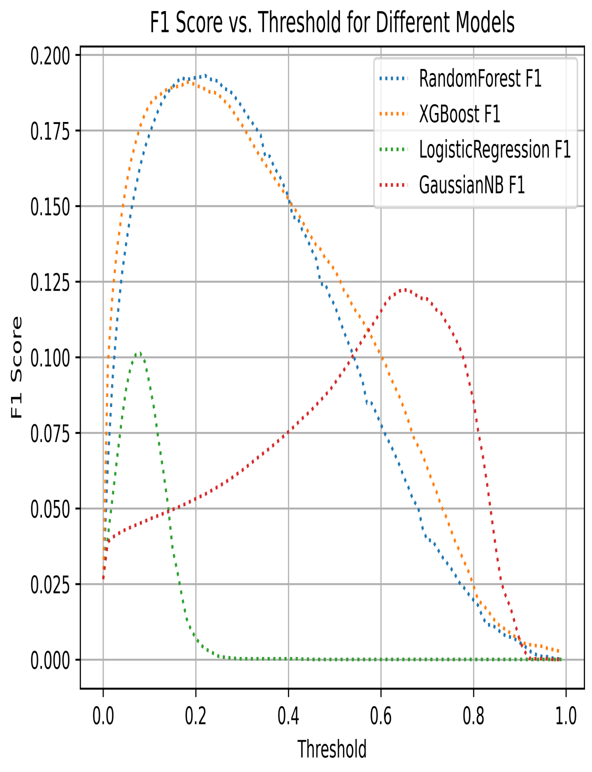
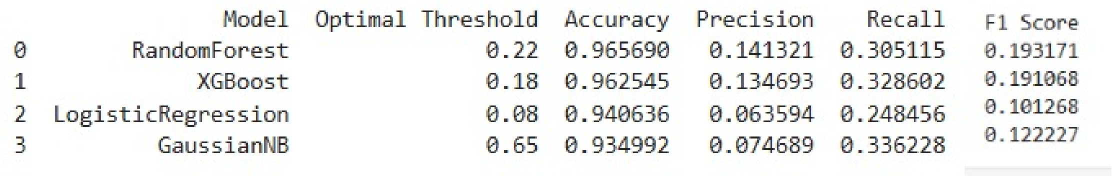


Figure 3: Thresholds for the optimal F1 score using each model

Figure 4: Statistics for each model fit on historical weather data

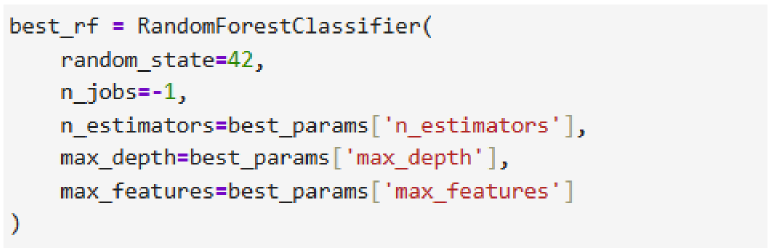


```
                 Model  Optimal Threshold  Accuracy  Precision    Recall   F1 Score
0         RandomForest               0.22  0.965690   0.141321  0.305115   0.193171
1              XGBoost               0.18  0.962545   0.134693  0.328602   0.191068
2   LogisticRegression               0.08  0.940636   0.063594  0.248456   0.101268
3           GaussianNB               0.65  0.934992   0.074689  0.336228   0.122227
```

Based on these results, we decided to go with a random forest model. Although the XGBoost and random forest models performed similarly, the random forest model had a higher F1 score as well as higher accuracy. Furthermore, we are more familiar with working with random forests. Given more time, we would explore the XGBoost model, especially since it has higher recall, which is desirable given that the cost of a false negative is high.

4.3 Hyperparameter Selection

After deciding on a random forest as the optimal model, we performed a grid search with 3-fold cross-validation to fine-tune model parameters. We used a random sample of 10% of the data due to storage limits. The parameters of interest were the number of estimators in the forest (50, 100, 200), the maximum depth of each tree considered (None, 10, 20, 30), and the number of features considered at each split (2, 3). Therefore, 24 possible combinations were considered. Using a decision threshold of 0.22, we selected the model with the highest F1 score. Ultimately, this was selected to be one with 200 estimators, a max depth of 30, and 3 features considered at each split. Figure 5 shows the trained random forest using this



```
best_rf = RandomForestClassifier(
    random_state=42,
    n_jobs=-1,
    n_estimators=best_params['n_estimators'],
    max_depth=best_params['max_depth'],
    max_features=best_params['max_features']
)
```

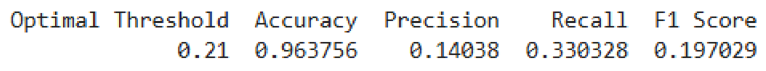
Figure 5: Code for Training the Optimal Random Forest

4.4 Results

After fitting the fine-tuned model to the training and testing data, we ended with Figure 6. Hyperparameter tuning was able to marginally improve F1 score (0.193 to 0.197) and recall

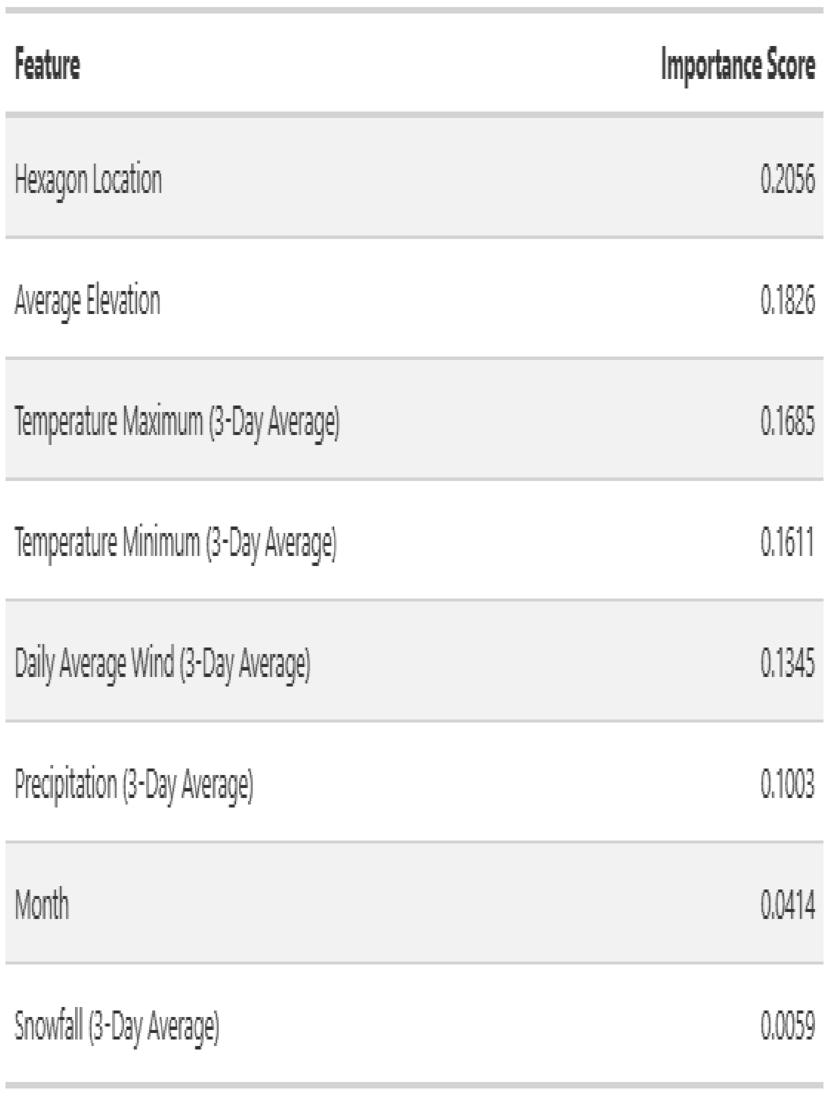
(0.305 to 0.330). Although these scores are still fairly low, this is somewhat expected due to the limit of our data and the variability of fire causes. Figure 7 shows a table of the most important features, showing that location, elevation, and temperature were seen as key predictors of wildfires.

Figure 6: Statistics for our Random Forest after Hyperpararmeter Tuning



```
Optimal Threshold  Accuracy  Precision    Recall  F1 Score
             0.21  0.963756    0.14038  0.330328  0.197029
```

Figure 7: Feature Importance



Feature	Importance Score
Hexagon Location	0.2056
Average Elevation	0.1826
Temperature Maximum (3-Day Average)	0.1685
Temperature Minimum (3-Day Average)	0.1611
Daily Average Wind (3-Day Average)	0.1345
Precipitation (3-Day Average)	0.1003
Month	0.0414
Snowfall (3-Day Average)	0.0059

After running the model, the output was saved as a parquet file and loaded later into the dashboard.

4.5 Challenges

One of the key challenges faced when running the model was the large amount of data we faced. Because we are dealing with both time and spatial data, we were dealing with a dataframe that has a row for every unique combination of dates from 2020-2025 and hexagon locations (of which there are 712). This gives a dataset with a total of 188,240,121 rows. This was too large to load into memory on our local devices, so we used The Catalyst. This is a high-performance computer that is reservable for students at the Iowa State Library. We loaded the data and were able to save the model output through OneDrive.

5 Dashboard

5.1 Exploration

We originally explored ArcGIS's online dashboarding tools due to its seamless integration with geospatial data. However, loading hexagonal maps with predictions for each date proved to be too large a task for the software to handle, leading to incredible slow loading times. As a result, we turned to Python's Dash library. This was able to handle large amounts of data and gave the option to add map layers easily.

5.2 Dashboard Creation

- Explored arcgis(slow, generally too limited in feature set)
- Chose dash because data was large and required cu
- stom real-time querying, needed efficient real-time map generation and rendering
- Stack: mixed python/r, dash, leaflet, containerized with docker, self-hosted
- Key visualization: fire risk probability, climate data

- Precomputed all values and plots needed for dashboard. The only realtime workload is selecting a single day's render fields.
- many many pictures

All of the values for the dashboard were precomputed from the model. For the colors of the map, output was normalized using the following formula:

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

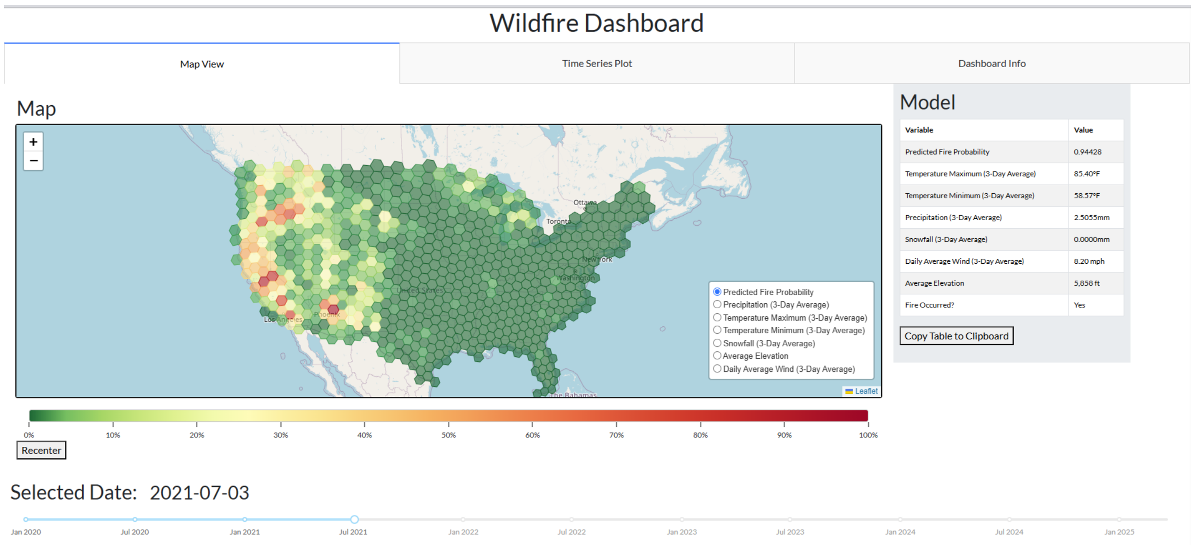
However, the actual predicted output and color bar on the map reverse this normalization so that the labels match the real predicted and weather values.

5.3 Interface

The dashboard has three main tabs. The map view allows users to set a date to view an overlay of wildfire predictions within hexagons on a map of the United States. The user can select the date displayed using a date slider at the bottom of the page. They can also select different variables in the tab menu on the map to get weather data for that date, allowing users to easily visualize weather patterns and areas of high fire probability. When viewing the map, users can click on a specific hex to get both model output and weather for that area on that selected date. Model output can be copied to clipboard for advanced analytics and reporting.

The time series plot was created in Python using Plotly and is hosted as an HTML file. This gives information on the number of fires occurring on a weekly, monthly, and yearly basis.

Finally, the dashboard info tab gives information about the underlying model and how to use the dashboard. It was created in R using Quarto and is also implemented as an HTML.



[✓ Predicted Fire Probability]
Precipitation (3-Day Average)
Temperature Maximum (3-Day Average)
Temperature Minimum (3-Day Average)
Snowfall (3-Day Average)
Average Elevation
Daily Average Wind (3-Day Average)

Selected Date: 2021-07-03

Model

Variable	Value
Predicted Fire Probability	0.94428
Temperature Maximum (3-Day Average)	85.40°F
Temperature Minimum (3-Day Average)	58.57°F
Precipitation (3-Day Average)	2.5055mm
Snowfall (3-Day Average)	0.0000mm
Daily Average Wind (3-Day Average)	8.20 mph
Average Elevation	5,858 ft
Fire Occurred?	Yes

Copy Table to Clipboard

Figure 8: An example wildfire prediction dashboard

5.4 Challenges

- Major issues and solutions (dash-leaflet immutability)

5.5 Availability/Source Code Location

All of the source code can be found at https://github.com/nathanrethwisch/FinalProjectDS4010A. The dashboard is currently hosted at https://wildfire.ddole.net/.

6 Conclusion

6.1 Summary

Using historical fire data from the United States Forest Service (USFS) and weather data from the National Oceanic and Atmospheric Administration (NOAA), we were able to create a random forest model and associated dashboard that predicted the probability of a fire occurring in a given area on a certain date. The dashboard allows users to explore dates and weather trends that lead to a high fire probability. This can be used for setting policy and garnering a greater understanding of the relationship between weather and wildfires.

6.2 Further Discussion

Given more time, we would aim to expand our analysis by incorporating additional datasets to enhance our predictive accuracy. We would include factors such as soil conditions and wind direction in specific areas as key predictors. Additionally, we would like to explore the probability of a fire occurring again in a region that has already had a recent wildfire. If we had access to greater computational resources, we would refine our predictions by using smaller hexagonal grids, allowing for more precise insights into distinct geographical climates. Moreover, the dashboard could be enhanced to support future fire prediction efforts. By using upcoming weather forecasts, we could provide a forecasted probability of fire occurrence in specific areas. These ideas lay the foundation for ongoing exploration and continued development of the dashboard.

In the future, this data should be moved into cloud object storage.

Sources

U.S. Congress Joint Economic Committee. 2023. "Climate-Exacerbated Wildfires Cost the u.s. Between $394 to $893 Billion Each Year in Economic Costs and Damages." https://www.jec.senate.gov/public/_cache/files/9220abde-7b60-4d05-ba0a-8cc20df44c7d/jec-report-on-total-costs-of-wildfires.pdf.