SMU Project 3

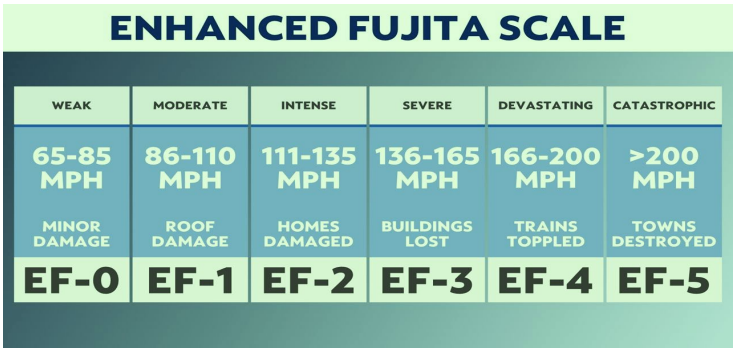
Creating an interactive Dashboard

Fabi Estrada
Uzor Francis
Ally Blitch
Ivette Reese

Table of Contents

I. Introduction

We chose the US tornado data because it had a wide range of data, and we were curious about how tornados are categorized. We discovered that the Enhanced Fujita scale (EF), which is used to rate tornados, relies on wind speed to determine a tornado magnitude rating, where 0 means a weak tornado. In contrast, 5 represents a catastrophic tornado, as shown below.



Tornadoes have significant impacts and consequences on human lives. They are powerful and destructive natural disasters that can cause loss of life, injuries, and extensive damage to infrastructure and property.

Our project aimed to uncover patterns, frequency, and severity of US tornadoes over the last seven decades, a data CSV file of tornado occurrences from 1950-2022 comprising 68,693 records and 27 columns. We want to tell the story about the trends in the number of tornadoes, magnitudes, or intensity by region of these natural hazards and how they have affected our lives.

II. Data Engineering

We started by conducting our exploratory data analysis using Jupyter Notebook. By using the df.info command we found out that we did not have any null values but dropped one identified duplicate, 756 magnitude ratings missing values, and 27,170 loss input missing values.

```
# Check data type and missing values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68693 entries, 0 to 68692
Data columns (total 27 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   om            68693 non-null  int64
 1   yr            68693 non-null  int64
 2   mo            68693 non-null  int64
 3   dy            68693 non-null  int64
 4   date          68693 non-null  object
 5   time          68693 non-null  object
 6   tz            68693 non-null  object
 7   datetime_utc  68693 non-null  object
 8   st            68693 non-null  object
 9   stf           68693 non-null  int64
 10  mag           67937 non-null  float64
 11  inj           68693 non-null  int64
 12  fat           68693 non-null  int64
 13  loss          41523 non-null  float64
 14  slat          68693 non-null  float64
 15  slon          68693 non-null  float64
 16  elat          68693 non-null  float64
 17  elon          68693 non-null  float64
 18  len           68693 non-null  float64
 19  wid           68693 non-null  int64
 20  ns            68693 non-null  int64
 21  sn            68693 non-null  int64
 22  f1            68693 non-null  int64
 23  f2            68693 non-null  int64
 24  f3            68693 non-null  int64
 25  f4            68693 non-null  int64
 26  fc            68693 non-null  bool
dtypes: bool(1), float64(7), int64(14), object(5)
memory usage: 13.7+ MB
```

```
# Check missing values by column
df.isna().sum()
```

```
om                0
yr                0
mo                0
dy                0
date              0
time              0
tz                0
datetime_utc      0
st                0
stf               0
mag             756
inj               0
fat               0
loss          27170
slat              0
slon              0
elat              0
elon              0
len               0
wid               0
ns                0
sn                0
f1                0
f2                0
f3                0
f4                0
fc                0
dtype: int64
```

We found the missing values then dropped them off.

```
# Drop rows with missing values
df.dropna(inplace=True)
```

```
# Another check missing values by column
df.isna().sum()
```

```
om              0
yr              0
mo              0
dy              0
date            0
time            0
tz              0
datetime_utc    0
st              0
stf             0
mag             0
inj             0
fat             0
loss            0
slat            0
slon            0
elat            0
elon            0
len             0
wid             0
ns              0
sn              0
f1              0
f2              0
f3              0
f4              0
fc              0
dtype: int64
```

We continued transforming the data and added three columns: region, months and num tornados to the database.

```
df["region"] = [state_regions[x] for x in df.st]
df.head()
```

	om	yr	mo	dy	date	time	tz	datetime_utc	st	stf	...	ns	sn	f1	f2	f3	f4	fc	region	month	num_tornados
0	192	1950	10	1	1950-10-01	21:00:00	America/Chicago	1950-10-02T03:00:00Z	OK	40	...	1	1	25	0	0	0	False	South	October	2499
1	193	1950	10	9	1950-10-09	2:15:00	America/Chicago	1950-10-09T08:15:00Z	NC	37	...	1	1	47	0	0	0	False	Southeast	October	1070
2	195	1950	11	20	1950-11-20	2:20:00	America/Chicago	1950-11-20T08:20:00Z	KY	21	...	1	1	177	0	0	0	False	South	November	904
3	196	1950	11	20	1950-11-20	4:00:00	America/Chicago	1950-11-20T10:00:00Z	KY	21	...	1	1	209	0	0	0	False	South	November	904
4	197	1950	11	20	1950-11-20	7:30:00	America/Chicago	1950-11-20T13:30:00Z	MS	28	...	1	1	101	0	0	0	False	South	November	2209

5 rows × 30 columns

III. Creating the SQLite File

On a second Jupyter Notebook, we loaded and stored the transformed data into an SQLite database. We ran the queries (sunburst query below), which automatically created an SQLite file. Then, the Flask app listened for the request, parsed out the input, and turned it into an SQL statement: select * from the database. Once we received the data, we created three visualizations based on research questions. We applied the Seaborn color palette crest to all charts.

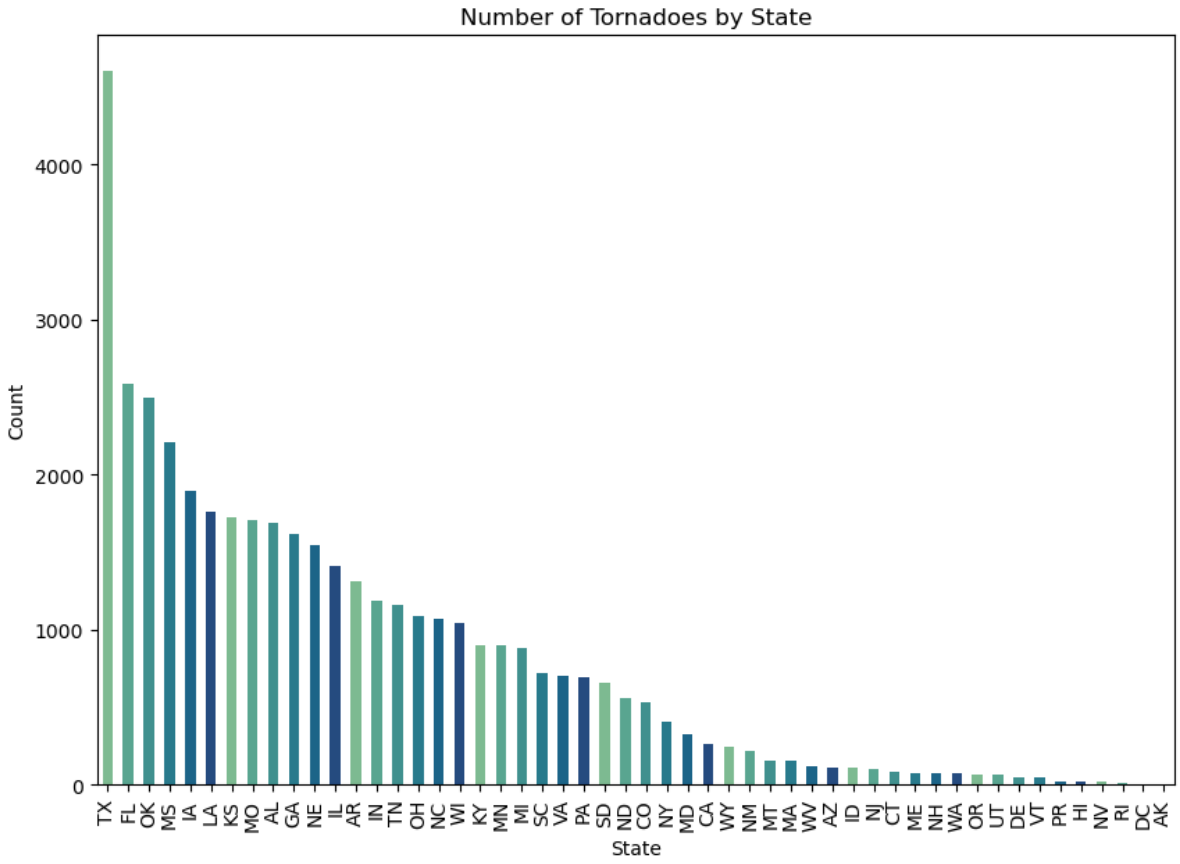
```
# For sunburst chart allow the user to select ALL or a specific state
if region == "All":
     where_clause = "1=1"
else:
    where_clause = f"region = '{region}'"

query = f"""
    SELECT
        region as label,
        "" as parent,
        count(*) as num_tornados
    FROM
        tornados
    WHERE
        {where_clause}
    GROUP BY
        region

    UNION ALL

    SELECT
        st as label,
        region as parent,
        count(*) as num_tornados
    FROM
        tornados
    WHERE
        {where_clause}
    GROUP BY
        st,
        region;
"""

df_sunburst = pd.read_sql(text(query), con=engine)
data_sunburst = df_sunburst.to_dict(orient="records")

df_sunburst.head()
```

	label	parent	num_tornados
0	Midwest		14617
1	Northeast		2040
2	South		14565
3	Southeast		8407
4	West		1887

IV. Data Visualization

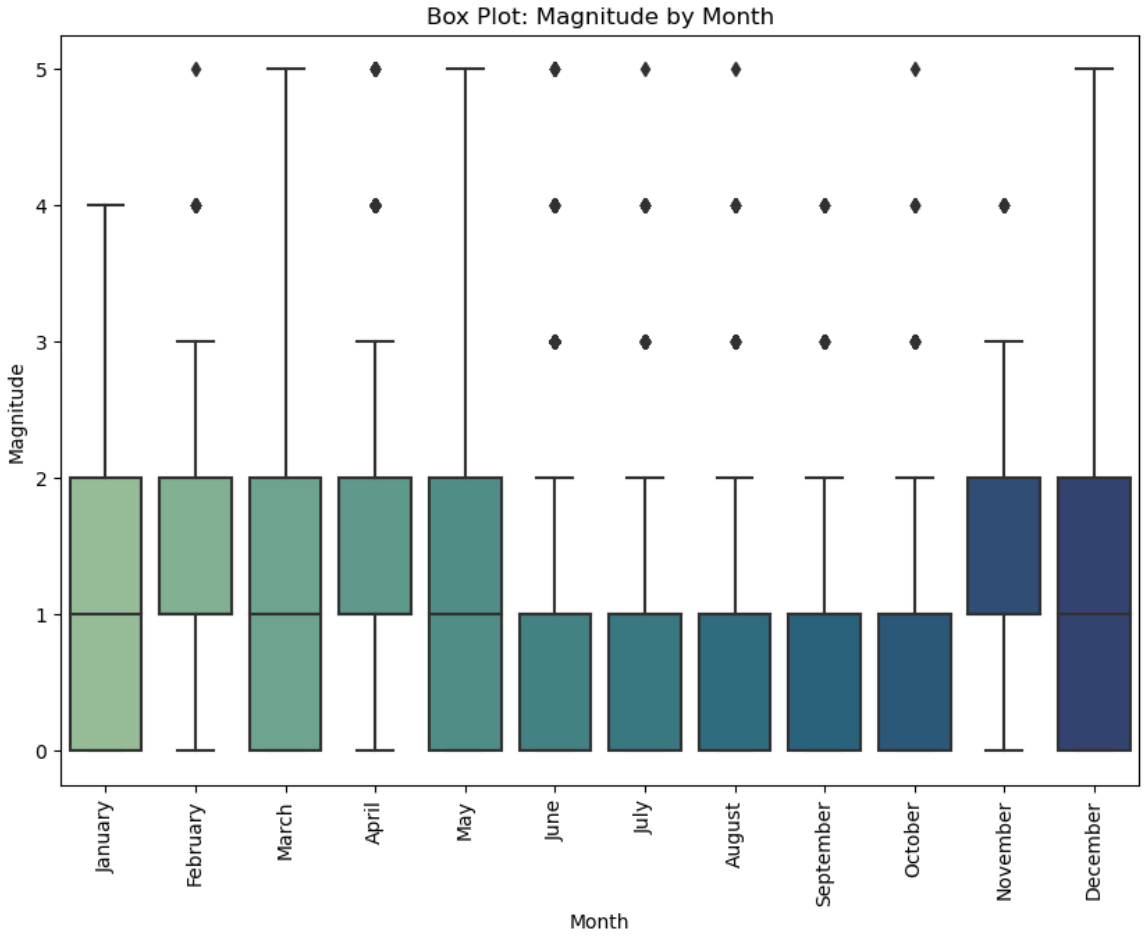
1. How many tornadoes occurred per state?

Using the matplotlib library, we created the above bar chart showing the number of tornadoes by state, from the state with the most tornadoes to the fewest. Texas had the highest number of tornadoes, 4,601, while DC and Alaska had the lowest number, 2 and 1, respectively.



2. In which month(s) do more tornados occur?

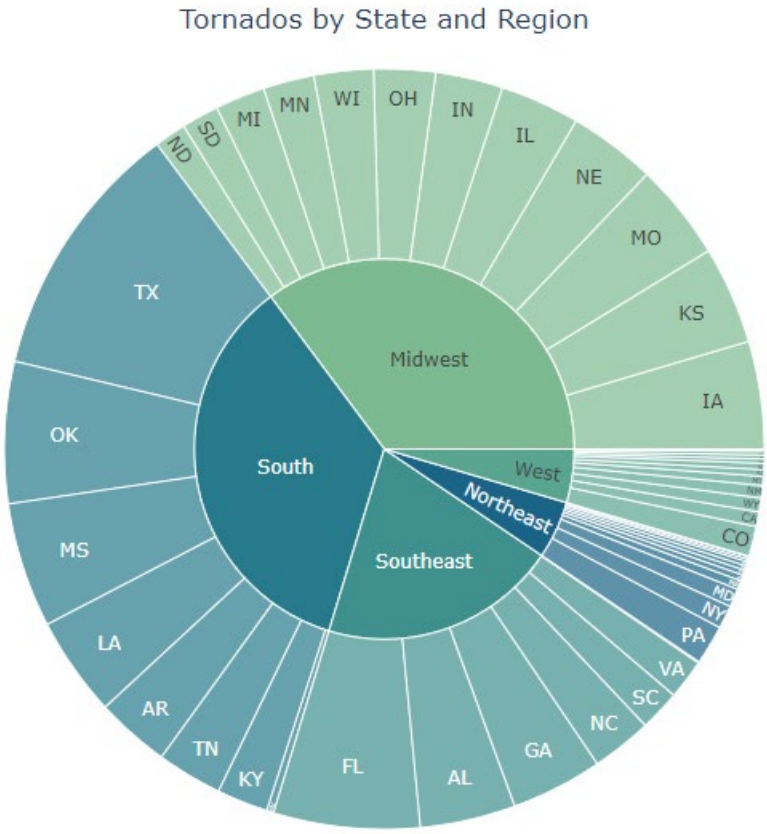
Using the matplotlib library, we also created a box plot chart that displays the number of tornadoes by month and magnitudes from 0 to 5.



Most tornados occurred during Winter (December through January) and Spring months (March through May). Since tornadoes are often associated with severe weather, severe weather conditions from June through October could result in higher tornado magnitudes, leading to outliers in the data.

3. How many tornadoes occurred per region?

Using the Plotly library, we created a sunburst chart that displays the number of tornadoes by region and state.

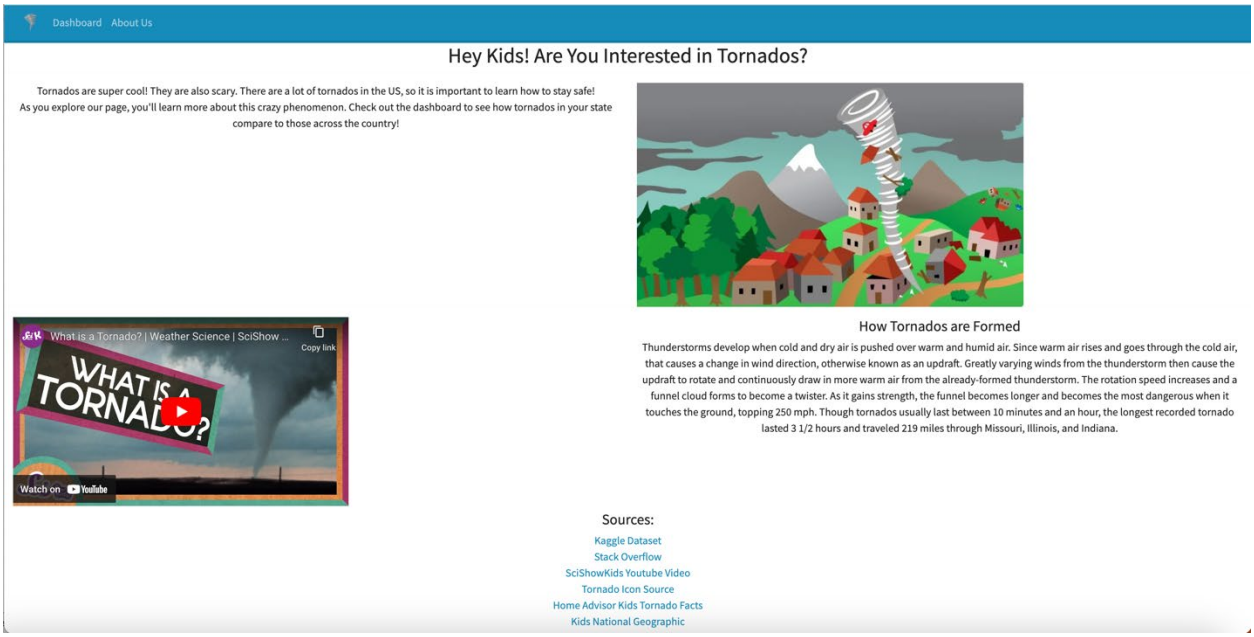


The highest concentrations of US tornadoes occurred in the Midwest region, with 14,617 tornados, followed by the South with 14,565 and the Southeast with 8,407.

V. Interactive Dashboard

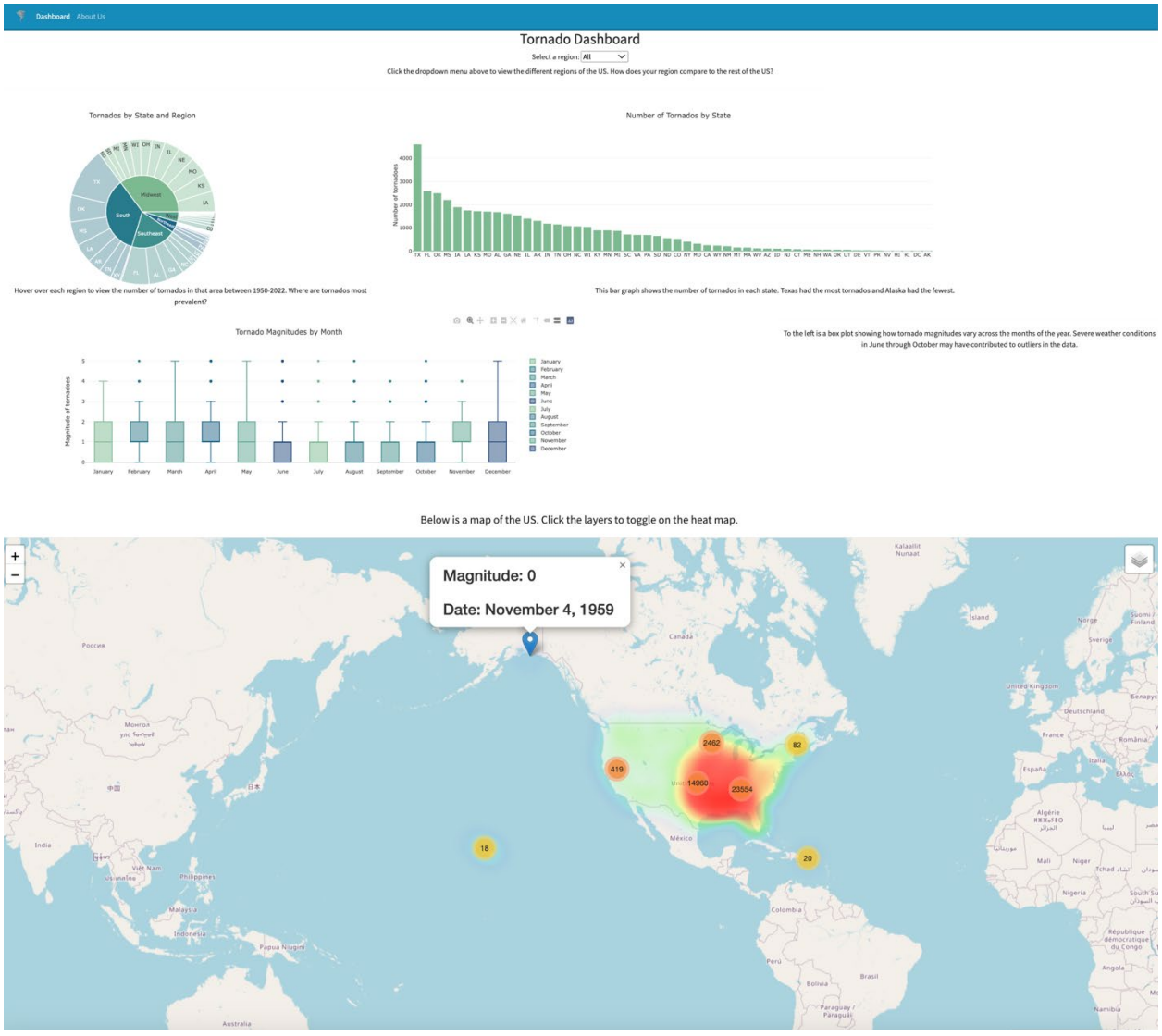
Using the JavaScript code and Plotly library, we created the dashboard. It consists of one dropdown menu that allows you to filter by region of the US.

The SQLite file uses Flask to serve data from the CSV database to one route and serve dashboard.html when the dashboard is visited. We created two more html files to create a home page (index.html) and an "About Us" page (about_us.html). The website opens to the home page first. We designed our website to be targeted toward children learning about tornados in school.

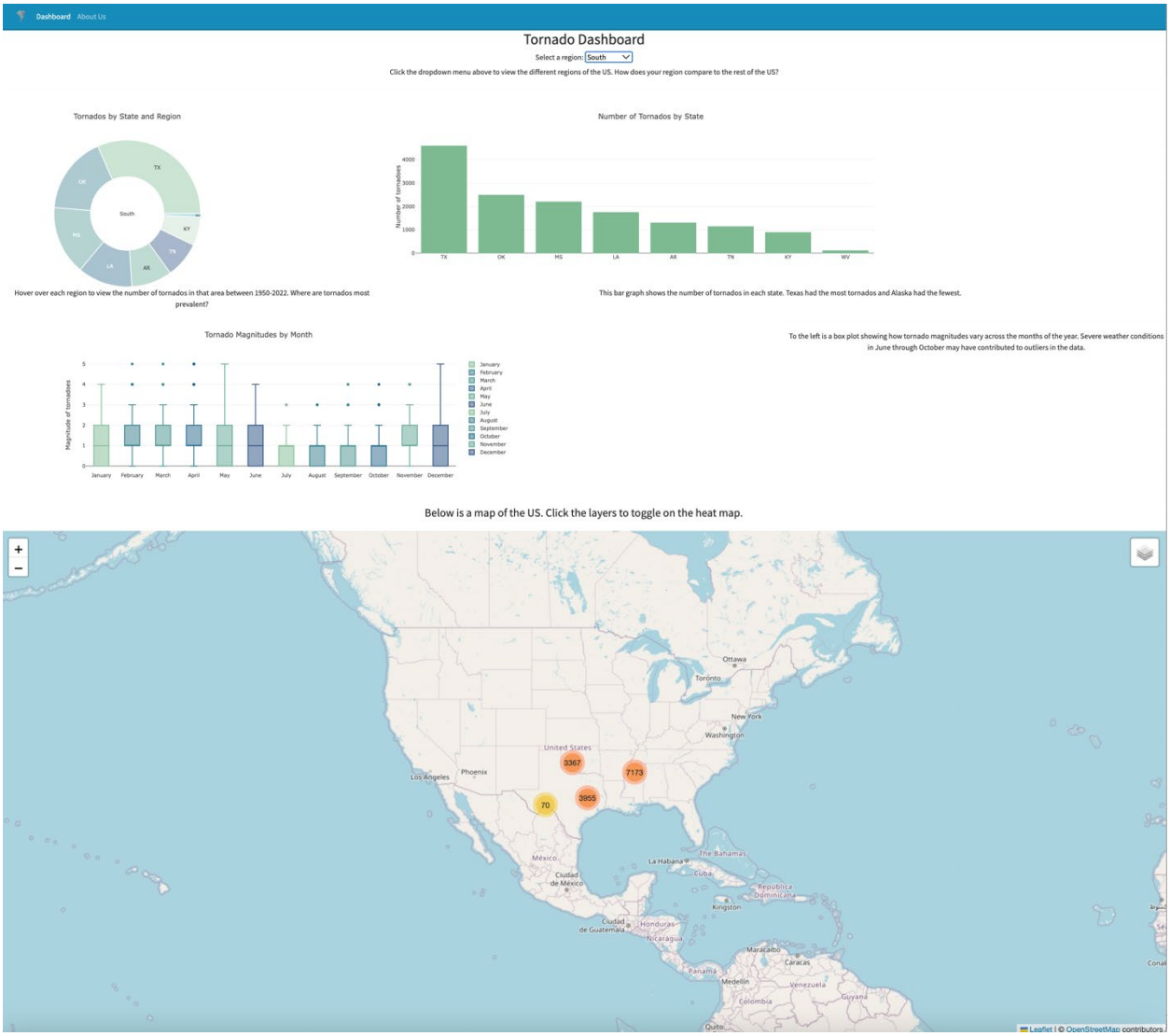


The dashboard is the second page on the website, followed by the “About Us” page. Our dashboard displays four views:

- A bar chart showing the number of tornadoes by state
- A box plot chart showing the tornado's magnitude per month
- A sunburst chart showing the number of tornados by US five regions
- A marker map showing the number of tornados by state and magnitude



Once a region is selected from the dropdown, all the visualizations update accordingly. For example, when “South” is selected, the dashboard updates to the following:



VI. Findings

1. Between 1950 and 2022, there were 68,693 tornados.

2. The highest concentrations of US tornadoes occurred in Texas, Florida, Oklahoma, Mississippi, Iowa, and Louisiana.

3. Texas had the highest concentration of tornadoes, with 4,601, while DC and Alaska had the lowest concentration of tornadoes, with 2 and 1, respectively.

4. The magnitude ratings show a strong upward trend, with 93% of the tornados reporting 0 to 2 magnitude ratings.

VII. Conclusions/Calls to Action

1. Texas is the most heavily hit by tornados compared to other states.

2. The US tornadoes (1950-2022) magnitudes' upward trend ratings were 0-2, meaning less severe tornadoes dominate our data.

3. Tornados occur in all US states, but the Midwest region is more prone to tornados. Tornadoes are most likely to happen in the Midwest.

VIII. Ethical considerations

Several ethical considerations were made to ensure the responsible use of data. First, our team ensured that all data sources used were obtained legally and with proper permissions. Additionally, we tried to minimize any potential bias from the data analysis. We carefully examined the data for any potential algorithmic bias and took steps to mitigate it. Overall, our team prioritized ethical considerations to ensure the responsible and ethical use of data throughout the project.

IX. Limitations & Bias

Accounting bias – according to the American Meteorological Society, the reporting of tornado occurrence is recorded at the county level. All tornadoes, regardless of size/EF rating, are counted. However, because of selective reporting, weak tornados with minimum impact/losses may go under-reported. Consequently, there has been some ongoing research on how tornadoes that are likely not to be measured can be better captured via supercells technology, an AI technology.

X. Future Work

With the advent of AI and other new technological breakthroughs in the 21st century, we believe AI will have a key role to play. It might be centered on using AI-based algorithms to perform regression analysis on Tornado datasets more effectively. This could take the shape of both or any of the following two frontiers of AI predictive power:

Estimating past tornado occurrence.

Predicting future tornado occurrence.

We predict AI capabilities could be used in several ways, including using an AI system that automatically identifies the effective drivers of tornado occurrences in particular regions of the State/Country and AI radar-based methods for capturing even the smallest tornadoes that usually go unnoticed according to major reports. Interestingly, with the adoption of AI, we can use automated linear and logistic regression to correctly identify some of our tornado features like duration, speed, and area covered.

In future projects, we would like to compare the starting and ending locations as our data included the longitude and latitude for both. Due to time constraints, in this project we solely looked at the starting latitude and longitude. We also think future analysis could include evaluating how climate change affects tornado occurrences. This could be done by comparing at average temperature of a year to tornado magnitudes or the number of tornados during that year.

Works Cited

https://www.kaggle.com/datasets/sujaykapadnis/tornados/data

https://www.weather.gov/oun/efscale

Horrific EF-5 tornado in Moore, Oklahoma: May 20, 2013 (youtube.com)

https://www.spc.noaa.gov/wcm/data/2008bams.pdf

https://www.gtri.gatech.edu/newsroom/new-approaches-including-artificial-intelligence-could-boost-tornado-prediction

Xpert Learning Assistant via BootcampSpot