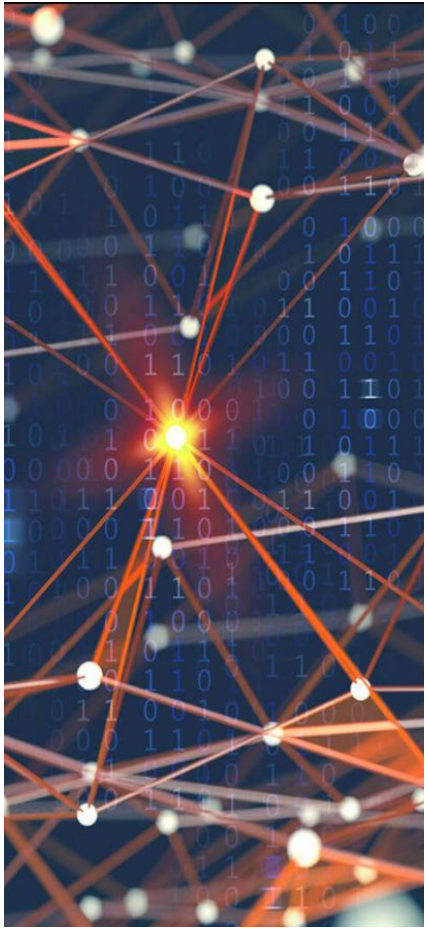
Wildfire and Weather Analysis

Group 2
Joe Sellner
Teena Stewart
Melissa Kamanzi
Derek Volz

Using Data to Predict Natural Disasters

Our group chose to do an analysis of weather and wildfire related data to see how the models would predict cause, count, and volume of natural disasters. We used machine learning to calculate predictions for number of wildfires, causes, number of tornadoes, and variance in weather patterns including min and max temperatures and departure from average.



Data Sources

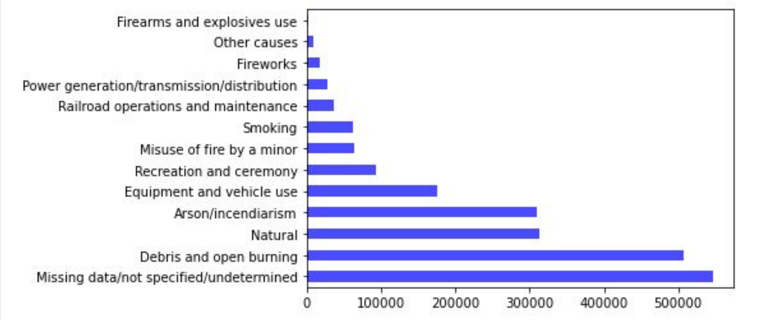
USDA Wildfire Data 1992 - 2013 (SQLite Database)

Weather Data 1992 - 2013 (API)

Tornado data 1950 - 2020 (Webscraping)

Fire Causes - Random Forest Classification

- Decision Trees are prone to Overfitting training data, Random Forest Classification alleviates that by testing multiple decision trees on the data to find the best fit
- One Hot Encoding was used to assign numerical values to each category, and to days of the week for the fire detection date data
- Wildfire data had 13 cause categories. Combining categories to decrease classification options will increase accuracy, but can take away from the overall story of the data



Fire Causes - Random Forest Classification

- Ran 2,000,000 predictions (original data was around 2,300,000 lines)
- ~58% accuracy using all 13 categories

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)
```

```
clf_rf = ske.RandomForestClassifier(n_estimators=50)
clf_rf = clf_rf.fit(X_train, y_train)
print(clf_rf.score(X_test,y_test))
```

```
0.5825834457585184
```

Fire Causes - Random Forest Classification

- ~82% accuracy when combining categories and using fewer choices (Accidental, Malicious, Natural, Unknown)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)
```

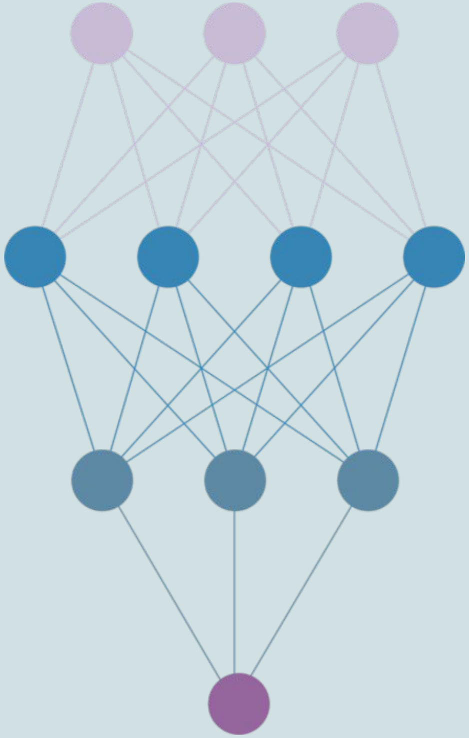
```
clf_rf = ske.RandomForestClassifier(n_estimators=50)
clf_rf = clf_rf.fit(X_train, y_train)
print(clf_rf.score(X_test,y_test))
```

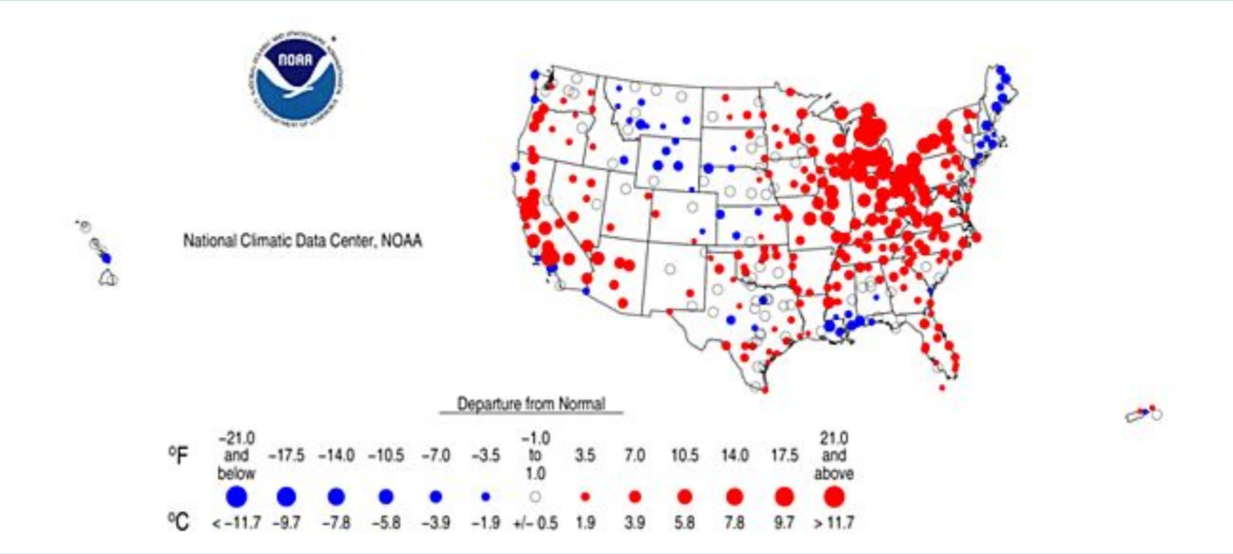
```
0.8265299827260015
```

- Given more time, creating trained models for individual states and comparing cause categories between them would have been interesting.

Fire Count - Tensorflow Keras Neural Network

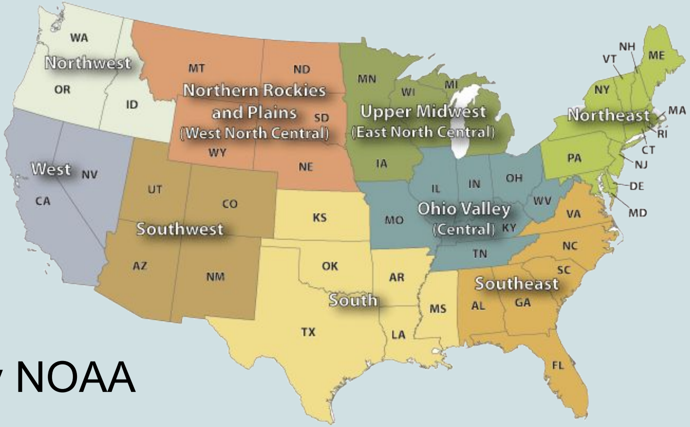
Can weather departure data explain/predict fire counts?

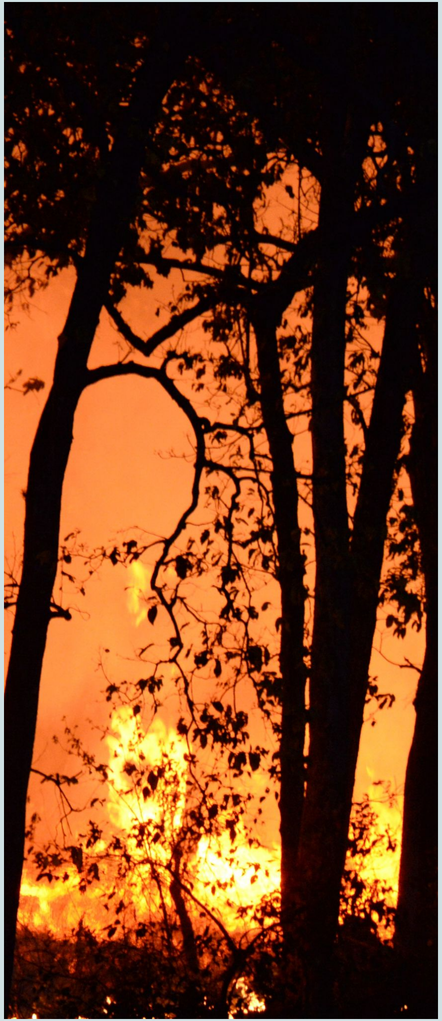




Fire Count - Data Input

- Departure from normal data 1992-2017:
  - Average Temperature
  - Precipitation
  - Min Temp
  - Max Temp
- Month
- Monthly wildfire count
  - 9 climate regions used by NOAA

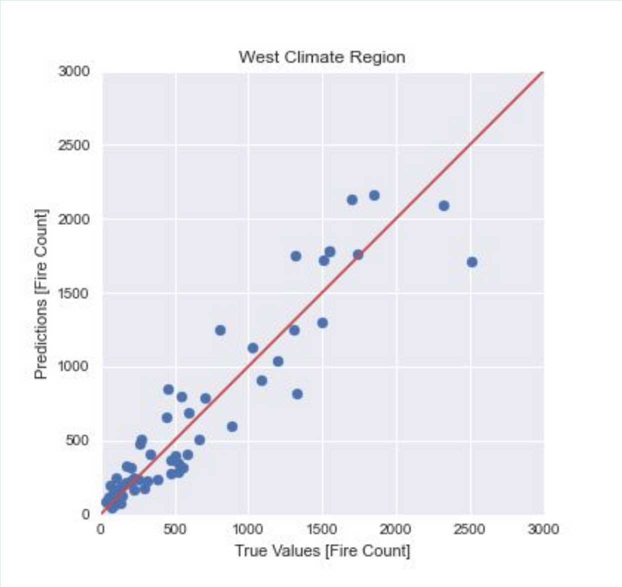


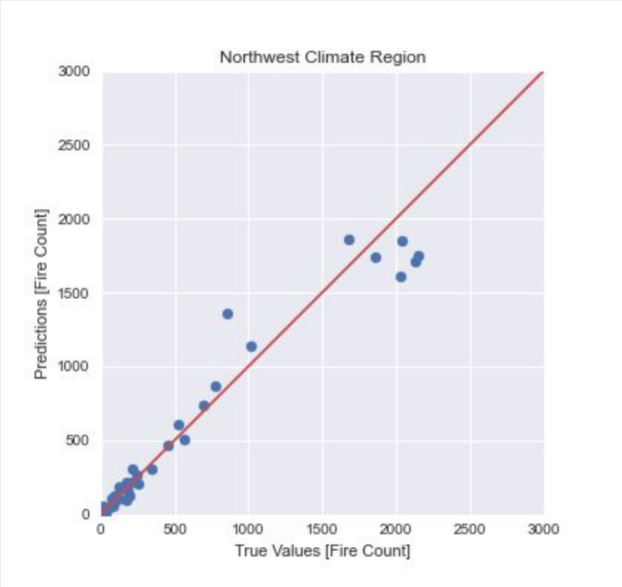


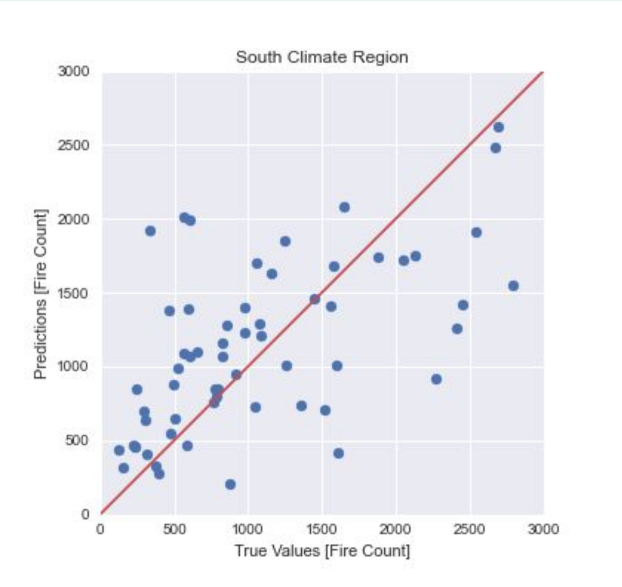
Fire Count - Model Construction

- Tensorflow Keras Sequential Neural Network
- Normalized input data with tensorflow preprocessing
- Random 80/20 train test split
- Separately trained model for each region
- 2500 epochs/iterations

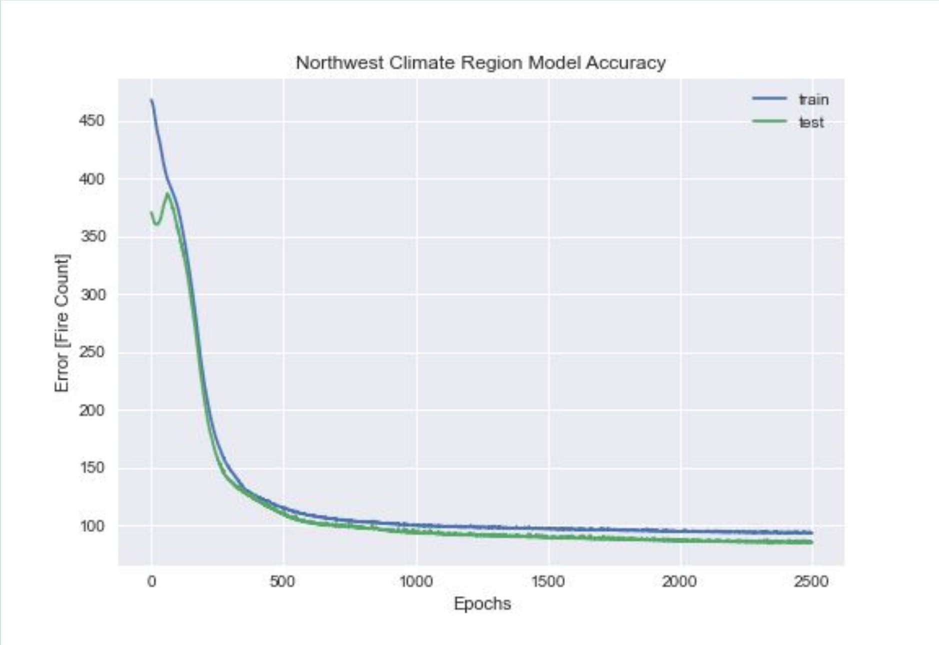
Fire Count - Results: Predicted vs Actual

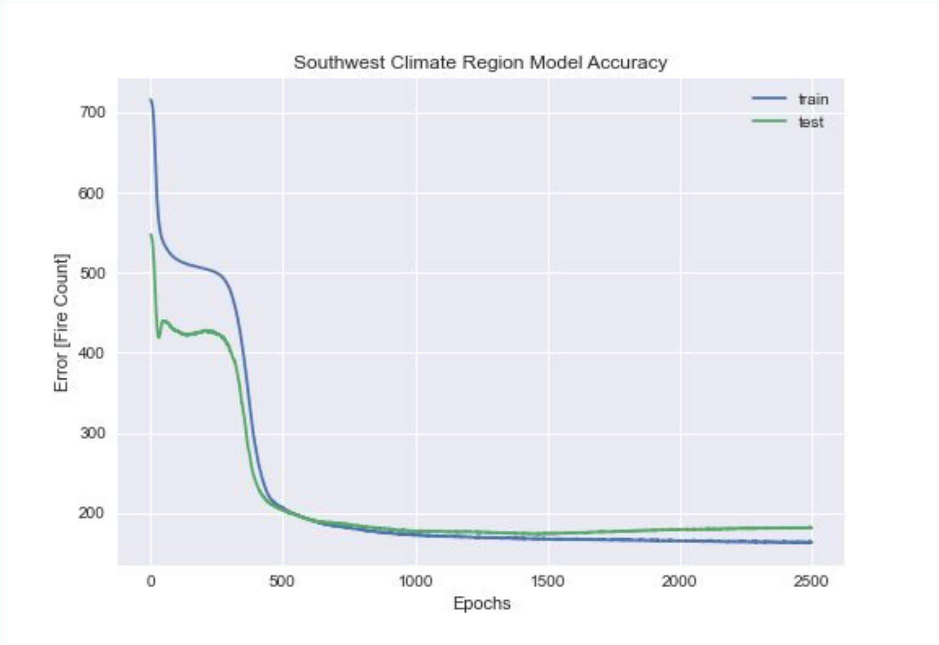






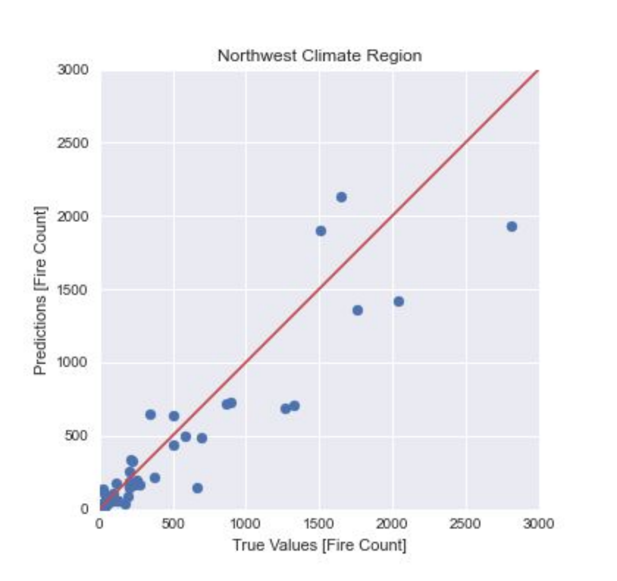
Fire Count - Training Accuracy

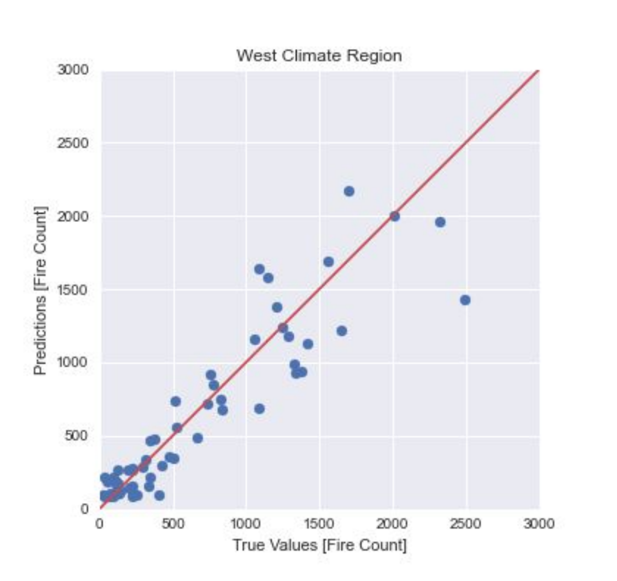


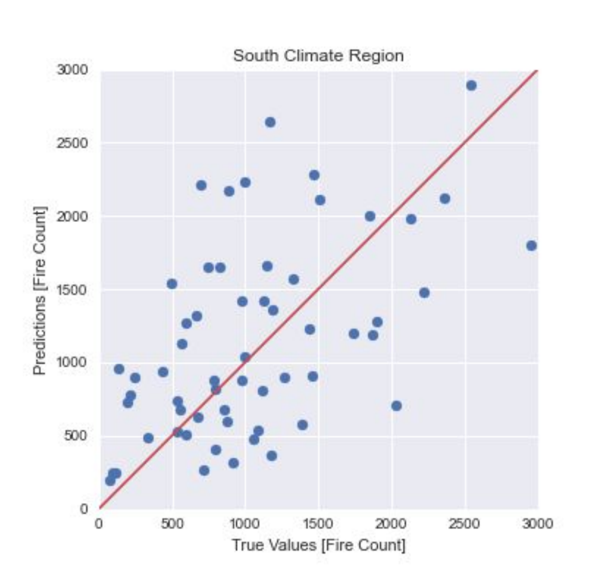


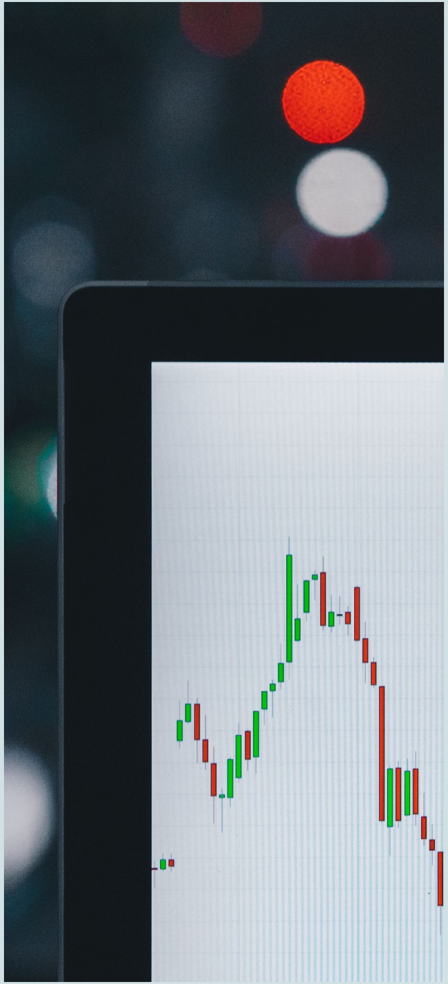
Fire Count - Future Prediction

Same data - time shifted by one month









Fire Count - Conclusions and Additional Analysis

Splitting by region was necessary to improve performance

Couldn’t find good info on importance of inputs to model

Additional temporal aspects are worth exploring

Departure data performed better than expected for some regions

Time permitting:

- More explanatory factors
- Increase spatial resolution
- Investigate poor performing regions
- Explore probabilities, confidence of prediction etc.

Predictions based on Min Temp, Max Temp and Precipitation

Data Cleaning

- Added a “Region_id” through a dictionary.
- Grabbed Minimum temperature, Maximum temperature, and Precipitation values from the weather table.
- Grabbed the fire count, date (converted into months) from the fire table.
- Grabbed the “Region_id” from both tables then later used it to merge the two tables

Merged table before predictions

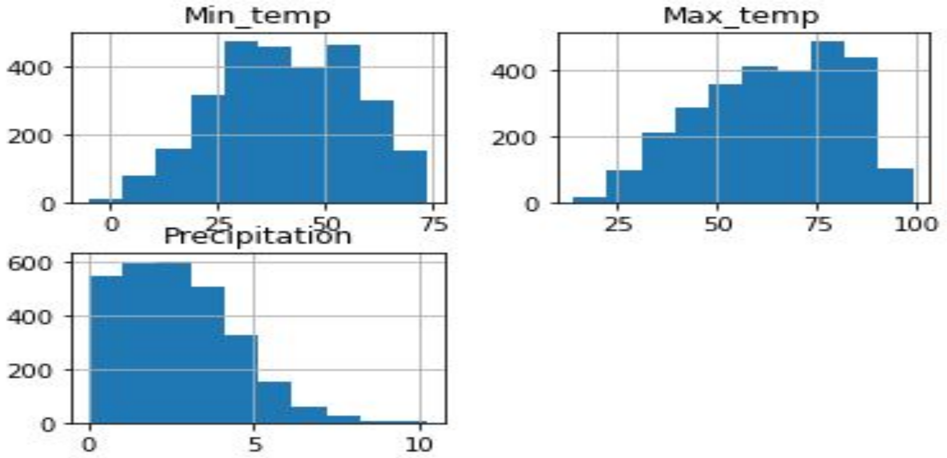
	Region_id	Date	Min_temp	Max_temp	Precipitation	Count	FPA_ID	Month
0	101	1992-01	14.6	32.5	2.47	86.0	86.0	01
1	101	1992-02	16.5	34.9	2.22	130.0	130.0	02
2	101	1992-03	20.2	39.8	3.70	283.0	283.0	03
3	101	1992-04	32.8	53.1	2.74	743.0	743.0	04
4	101	1992-05	41.8	67.4	2.77	830.0	830.0	05

Linear Regression - Making predictions

- Assigned values to X and y
- Removed outliers to decrease variability in the data.
- Reassigned values to X and y after outliers were removed.
- Split the data using *train_test_split*.
- Created the linear regression model.
- Printed predicted values
- Cleaned up final “predictions” DataFrame

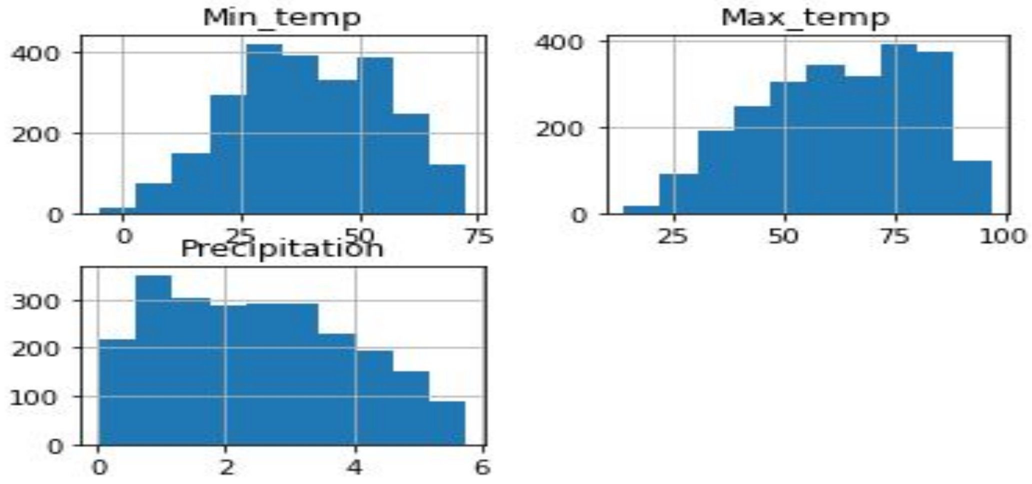
Min_temp, Max_temp and Precipitation variables - Histograms before removing outliers

```
23]: # Plotting all Values included in features_df
     features_df[["Min_temp", "Max_temp", "Precipitation"]].hist(bins=10)

23]: array([[<AxesSubplot:title={'center':'Min_temp'}>,
             <AxesSubplot:title={'center':'Max_temp'}>],
            [<AxesSubplot:title={'center':'Precipitation'}>, <AxesSubplot:>]],
           dtype=object)
```



Min_temp, Max_temp and Precipitation variables - Histograms after removing outliers

```
[31]: df_Nooutlier[["Min_temp", "Max_temp", "Precipitation"]].hist(bins=10)

[31]: array([[<AxesSubplot:title={'center':'Min_temp'}>,
              <AxesSubplot:title={'center':'Max_temp'}>],
             [<AxesSubplot:title={'center':'Precipitation'}>, <AxesSubplot:>]],
            dtype=object)
```



Tornado count - Data

Data obtained from - www1.ncdc.noaa.gov

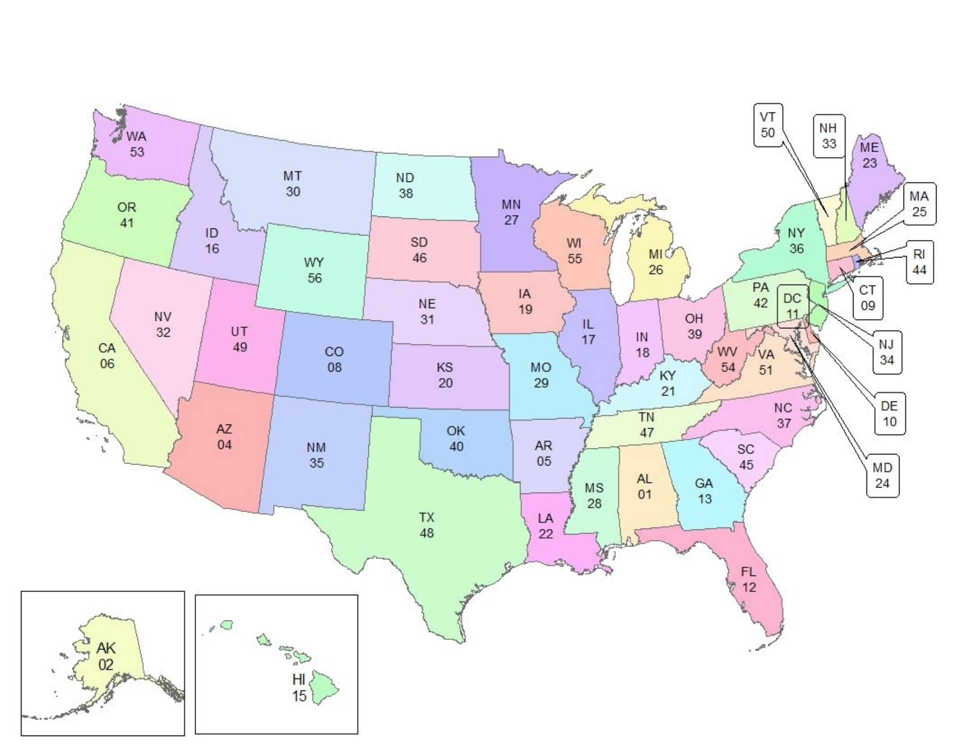
Web scraping and BeautifulSoup

- Data from 1950 to 2021 where event_type = Tornado

```
names = soup.find_all('a',{"class":""})
stormevent_details_all = []

# For loop to get all the names
for name in names[7:-2]:
        #print(name)
    #getting only the files with storm events
    if name.attrs['href'].startswith('StormEvents_details'):
        filename = name.attrs['href']
        storm_url = link+filename
        #print(storm_url)

        iter_csv = pd.read_csv(storm_url, compression='gzip', iterator=True, chunksize=1000)
        stormevent_details_all.append(pd.concat([chunk[chunk['EVENT_TYPE'].map(lambda x: x.lower())
                                                        == 'tornado'] for chunk in iter_csv], ignore_index=True))

combined_tornado_df = pd.concat(stormevent_details_all)
```

Tornado Count: Data

- Data from 1950 - 2021
  - Month
  - Time of day (military time)
  - State_fips (Geographical areas)
    - Count of tornados



** Note: FIPS codes are numbers which uniquely identify geographic areas

TensorFlow

- Random 80/20 train test
- Ran for each state’s FIPS code
- Iterations of 100

Results saved to csv file for Tableau visualization

Tornado Count Analysis

The data I used seemed not effective in predicting

- Used non-representative data of weather

Most the predictions were lower than actual

Time permitting:

- Look at “Tornado EF Scale”
- Build a model against weather data
    - Temperature
    - Precipitation
    - etc