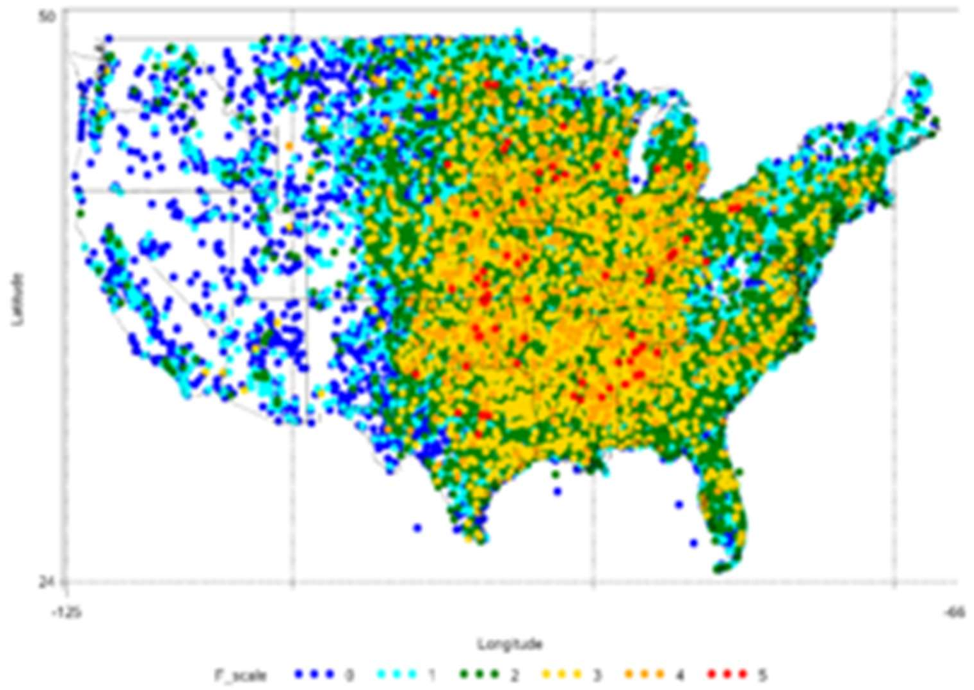
Travis Martin

Final Report: Tornado Impact Analysis

Problem Statement

Of all the forces mother nature can bring to bear on mankind, one of the most brutal and unpredictable is the tornado. A tornado is a violently rotating column of air connecting a thunderstorm with the ground, but that description potentially masks how destructive these weather events can be. Even the smallest and weakest tornadoes can uproot trees and toss vehicles around. When you move to the other end of the scale, 300+ mph winds, 2+ mile diameters, and 60+ mile ground tracks can create unthinkable devastation across vast swaths of land, especially in more populous areas, where tens or hundreds of homes and businesses can be completely leveled in a matter of minutes.

Tornadoes occur all over the world, but the United States endures more of them than any country on Earth, nearly 1,200 per year. For context, the next country on the list is Canada, which averages 60-100 tornadoes each year. As such, tornadoes are at least a minor consideration for all aspects of daily life in much of the United States. The central and southeastern portions of the country, in fact, are so prone that the region is colloquially known as "Tornado Alley".



All tornadoes in the lower-48 of the US 1950-2013, colored by F-Scale. Source: NOAA Storm Prediction Center

While property damage is essentially unavoidable when a tornado strikes a populated area, the same is not necessarily true when it comes to deaths and injuries to the humans unlucky enough to be caught in its path. Meteorologists can spend their entire careers studying these destructive storms to improve warning lead-time and lessen the likelihood of loss of life. Improvements in radar technology, the proliferation of trained "Storm Spotters", the official watch/warning system, and most recently, the practice of sending emergency alerts to cell phones in a tornado's path have all led to a gradual increase in the amount of advance notice that citizens receive ahead of the tornado's arrival, saving countless lives.

One of the agencies of Meteorologists studying these storms, NOAA (the National Oceanic and Atmospheric Administration), has diligently collected data on all the tornadoes impacting the United States in the last 70 years. But can this basic data tell us anything about whether a given tornado, (defined by measurements like width, start/end location, and distance covered) is more or less likely to cause injuries or deaths? This is the question we'll seek to answer, and we'll do so by feeding our preprocessed tornado data into a series of predictive Machine Learning models.

NOAA Dataset

The raw NOAA dataset initially contained and 66,389 rows and 29 columns, but it would require significant rework to prepare it for straightforward analysis and ML modeling. Inspecting the first ten rows of the dataset reveals the following:

First is the om column, which is a tornado identifier. This is a necessary identifier for later analysis, however from consulting the documentation, I learned that there is not a 1:1 relationship between unique om numbers and rows. These om numbers reset each year, and further, if a tornado passes through more than 4 counties, the 5th/9th/etc. counties create a new row with the same om value. Similarly, if a tornado crosses into any additional states, each state will have its own row. This multi-row issue affected nearly 1900 of the tornadoes in the dataset.

The next 4 columns are all date related, representing year, month, day, and aggregated date.

This is followed by:

- Time data (time and time zone)
- Geographic data (state id, state number)
- The number of that tornado in that state for that year
- The magnitude of the tornado (on the F or EF-scale depending on year)
- Human toll (fatalities, injuries)
- Economic damage (property damage, crop damage)
- Geographic track (starting latitude and longitude, ending latitude and longitude, and length of the track in miles)
- Width of the tornado in yards

The last 7 columns all relate to the geographies (counties, states) impacted by the tornado's track, and are not easy to parse without taking a hard look at the documentation.

ns, sn, and sg create a coded set of three numbers that represent the following:

- ns = number of states affected by the tornado
- sn = the state number for the specific row
- sg = the segment track number for the specific row

As mentioned above, there are 3 different scenarios that would produce multiple rows with the same om identifier:

- A tornado moves through more than 4 counties
- A tornado moves through 2 or more states, but never through more than 4 counties in any state
- A tornado moves through 2 or more states, also moving through more than 4 counties in one or more of the states

Tied to these three coded columns are the 4 county FIPS code slots: f1, f2, f3, and f4. These are populated with either the unique FIPS code for an affected county, or blanks if the end of the county list has been reached.

Data Wrangling

The dataset was loaded into a Pandas dataframe for the Data Wrangling process. As mentioned above, the most critical (and labor-intensive) change which needed to be performed was to ensure that each row represented the full complement of data for an individual tornado and possessed a unique identifier. This required expanding the number of county columns per row to be equal to the maximum number of counties listed for a single tornado. There is a tornado with 15 counties listed, but several counties are listed more than once (meaning a tornado passed back and forth across a county line). When these duplicates were removed, the list consisted of 11 unique entries. As such, the dataset was expanded to contain 11 county FIPS columns, thus consolidating multi-row instances down to a single row.

Another significant area of concern for the fidelity of this dataset is an apparent disconnect in how Property Damage is handled. A review of summary statistics for this variable shows a max value of 1.55 *billion* with a 75th-percentile value of 4, which implies a significant mismatch. Again, the documentation makes clear the issue:

> *Estimated property loss information - Prior to 1996 this is a categorization of tornado damage by dollar amount (0 or blank-unknown; 1=less than $50, 2 =$50-$500, 3 =$500-$5,000, 4 = $5,000-$50,000, 5 = $50,000-$500,000, 6 = $500,000-$5,000,000, 7 = $5,000,000-$50,000,000, 8 = $50,000,000-$500,000,000, 9 = $500,000,000-$5,000,000,000). From 1996,*

this is tornado property damage in millions of dollars. Note: this may change to whole dollar amounts in the future. Entry of 0 does not mean $0.

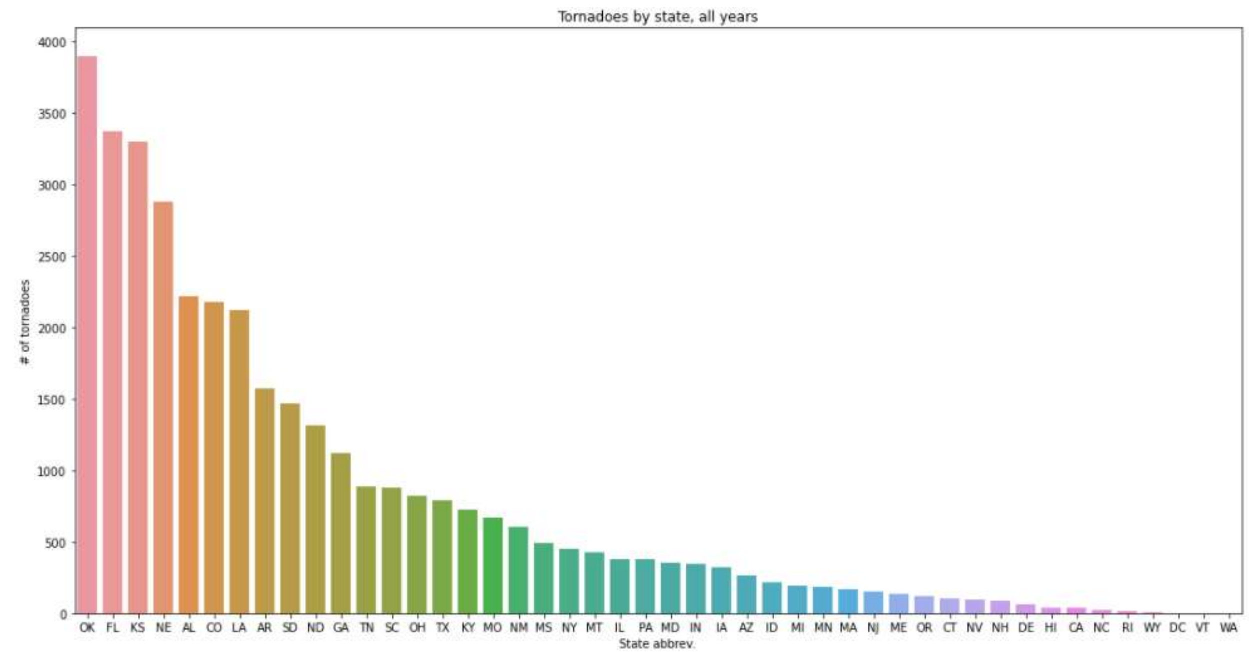
This explains the two different formats for this property loss data, but there's also an implied third. A maximum of 1.5B only makes sense if that amount is in whole dollars as opposed to millions of dollars. (For context, Hurricane Katrina "only" caused $161B in damage). Upon further investigation, this proved to be a miscoded value in whole $ instead of $M. I chose to re-classify the post-1996 $M damage amounts via the pre-1996 integer scale to marry those two segments.

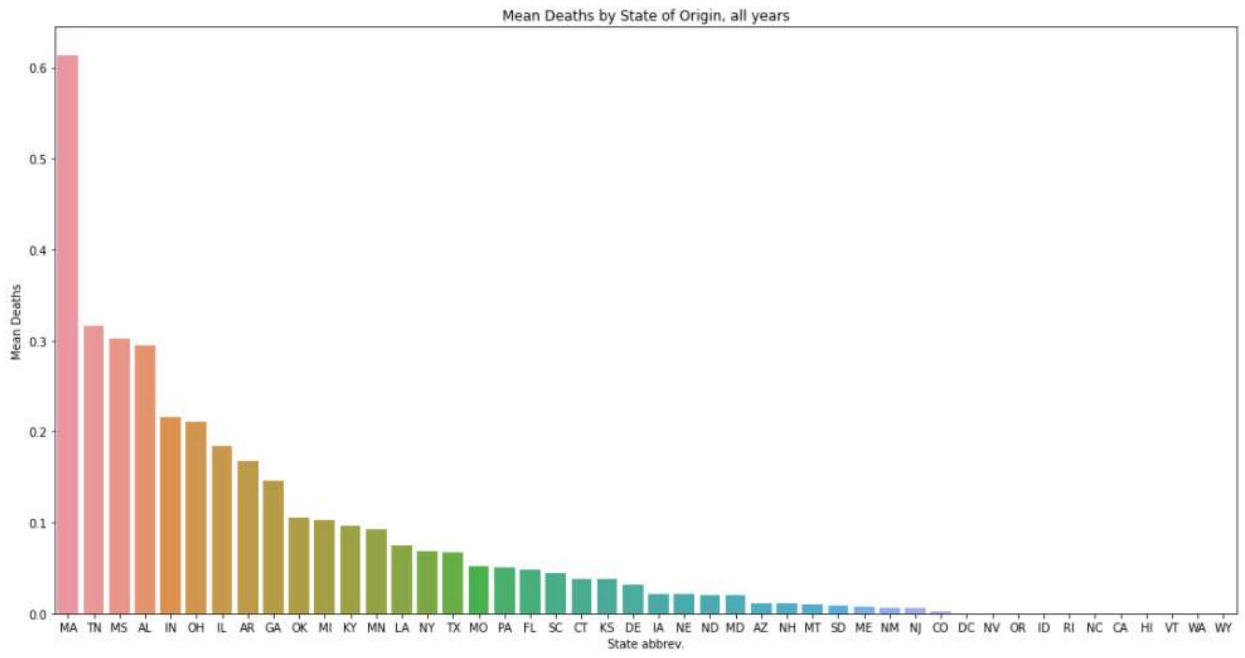
There were also 298 tornadoes for which the magnitude was unknown (approximately 0.4% of the dataset), and none of these records contained any data in the impact categories I was primarily concerned with (deaths/injuries/property loss), and thus they were simply removed.

It’s intuitive that tornadoes striking in more densely populated areas are more likely to cause deaths and injuries, so I decided to marry the base dataset with another that provides Land Area and Population Density for each county (and its associated FIPS code) in the US. This accessory dataset came from the most recent US Census Data on www.Data.gov. Population Density and Land Area was calculated for the full tornado track for each row, and then was joined to the base dataset with a simple Pandas merge.

Exploratory Data Analysis

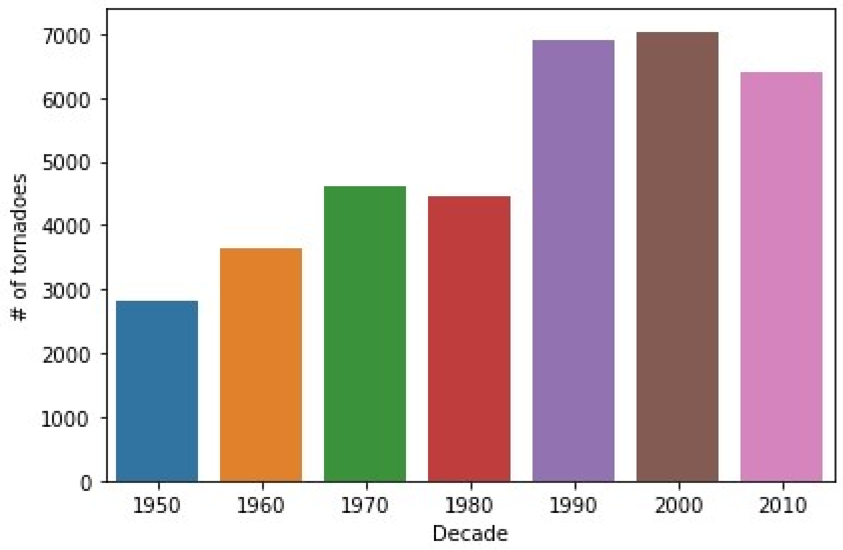
The primary purpose of the Exploratory Data Analysis step was to take a more focused look on each of the potential predictive variables, and look for trends or insights which might lead to a more predictive model. This is primarily done through the creation of both data tables and data visualizations using both Matplotlib and Seaborn Python libraries. One of the first thing that I was interested in exploring further was whether the states producing the greatest number of tornadoes also produced the most *deaths and injuries* from tornadoes.



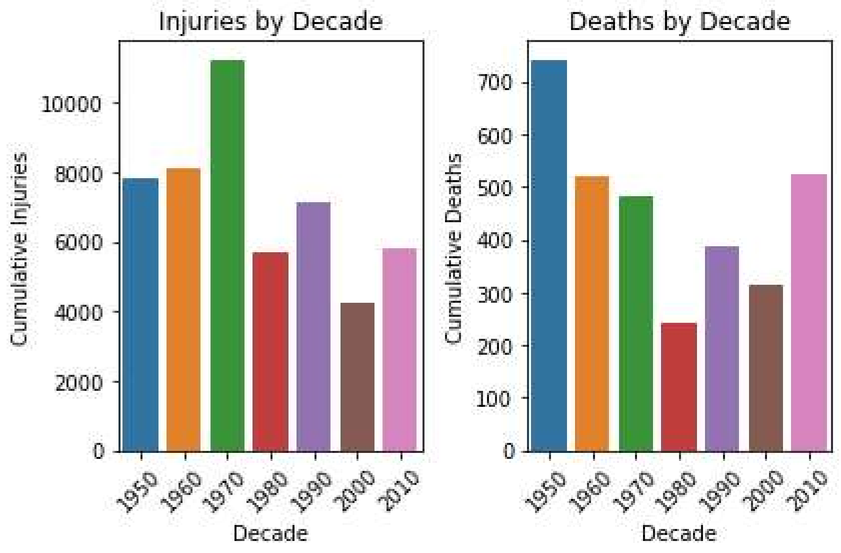


Upon inspection of the data, it's revealed that the top states mean number of deaths per tornado are quite different than the top states for number of tornadoes overall. In fact, Alabama is the only state which appears in the top 5 of both lists. In general, the states with higher mean deaths are smaller and further East than the plains states that dominate in the total number of tornado touchdowns. This seems feasible, as the United States tends to be sparsely populated in those central plains states and gets denser as you move toward the eastern seaboard.

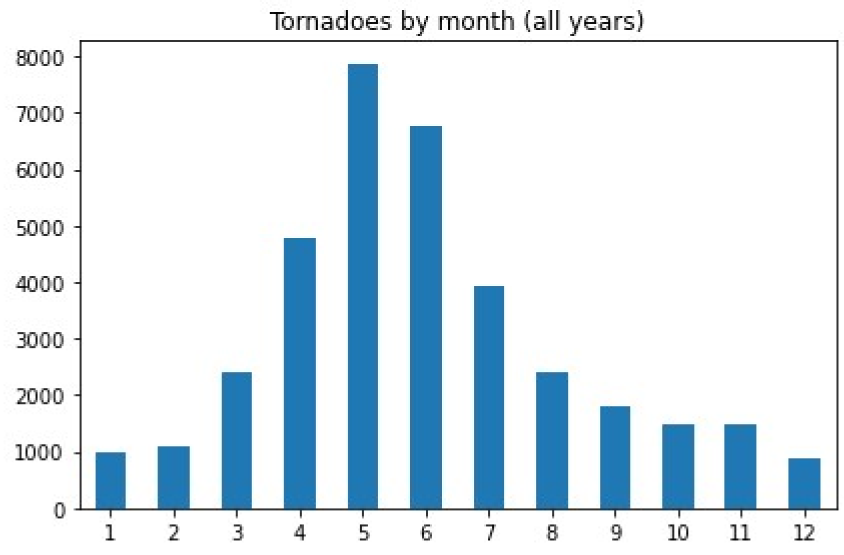
Another potential trend I wanted to investigate was whether there had been any change in the prevalence of tornadoes over time. As shown below, there has been a significant increase over time, though it appears to have plateaued slightly since 1990.

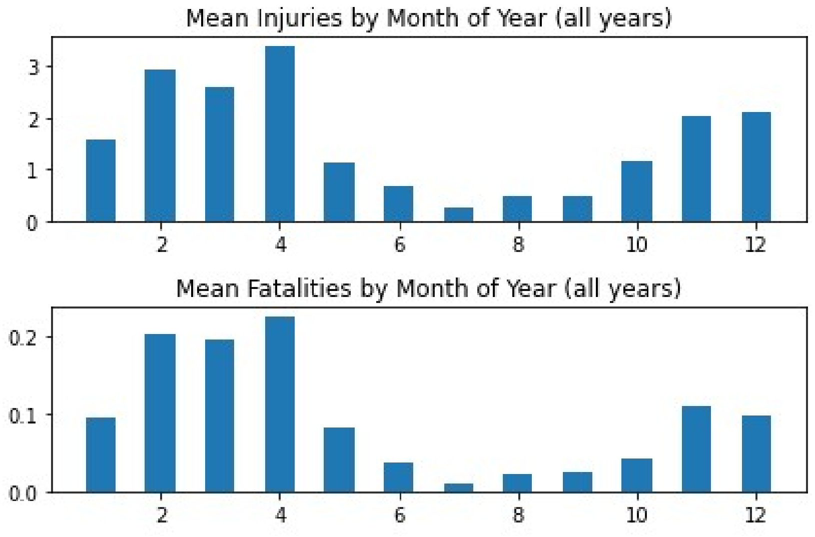


But what about those improvements in the tornado prediction and warning systems? Have those resulted in a measurable decrease in deaths and injuries? The answer appears to be yes, though the 2010s seem to show a significant uptick to the general decline in human toll from the 1950s through the 2000s.

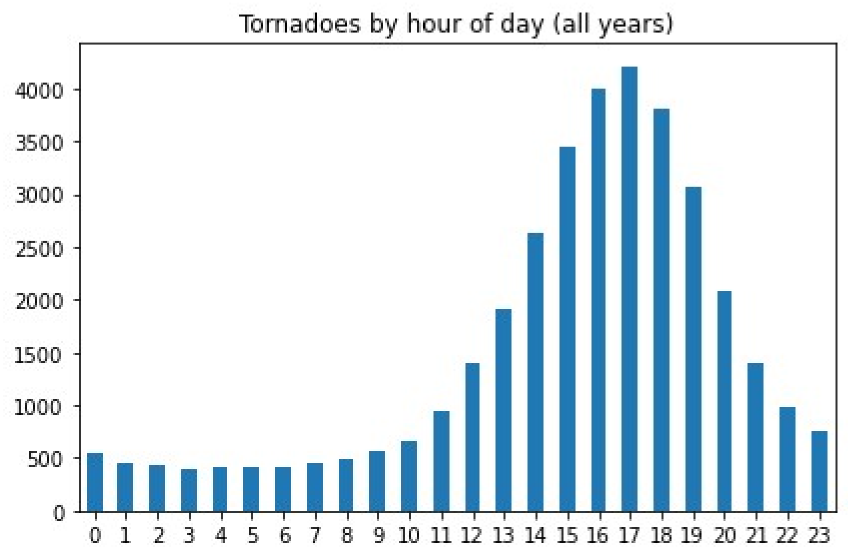


Is there a seasonality to tornadoes? This is clearly borne out by the data, as the vast majority occurred in the period between April and July. But it's also clear that the deadliest tornadoes seem to occur in the non-peak periods, when perhaps tornado preparedness is not top-of-mind.

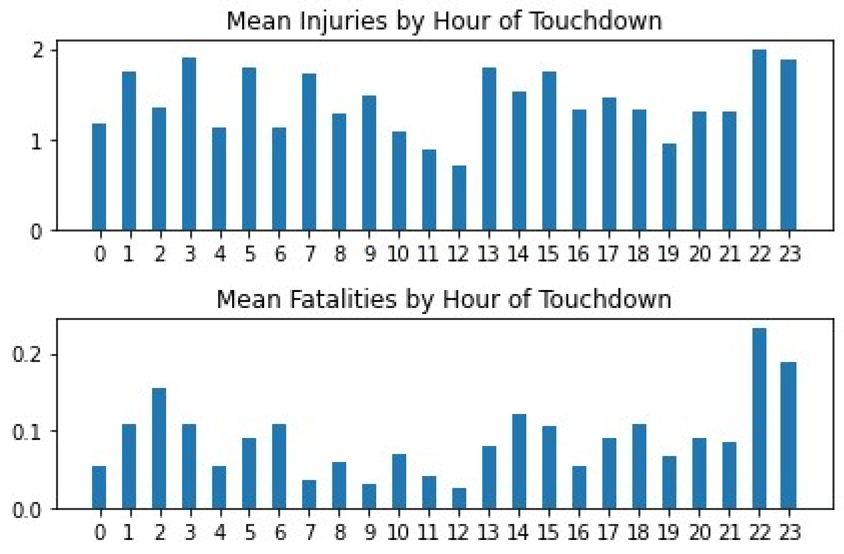




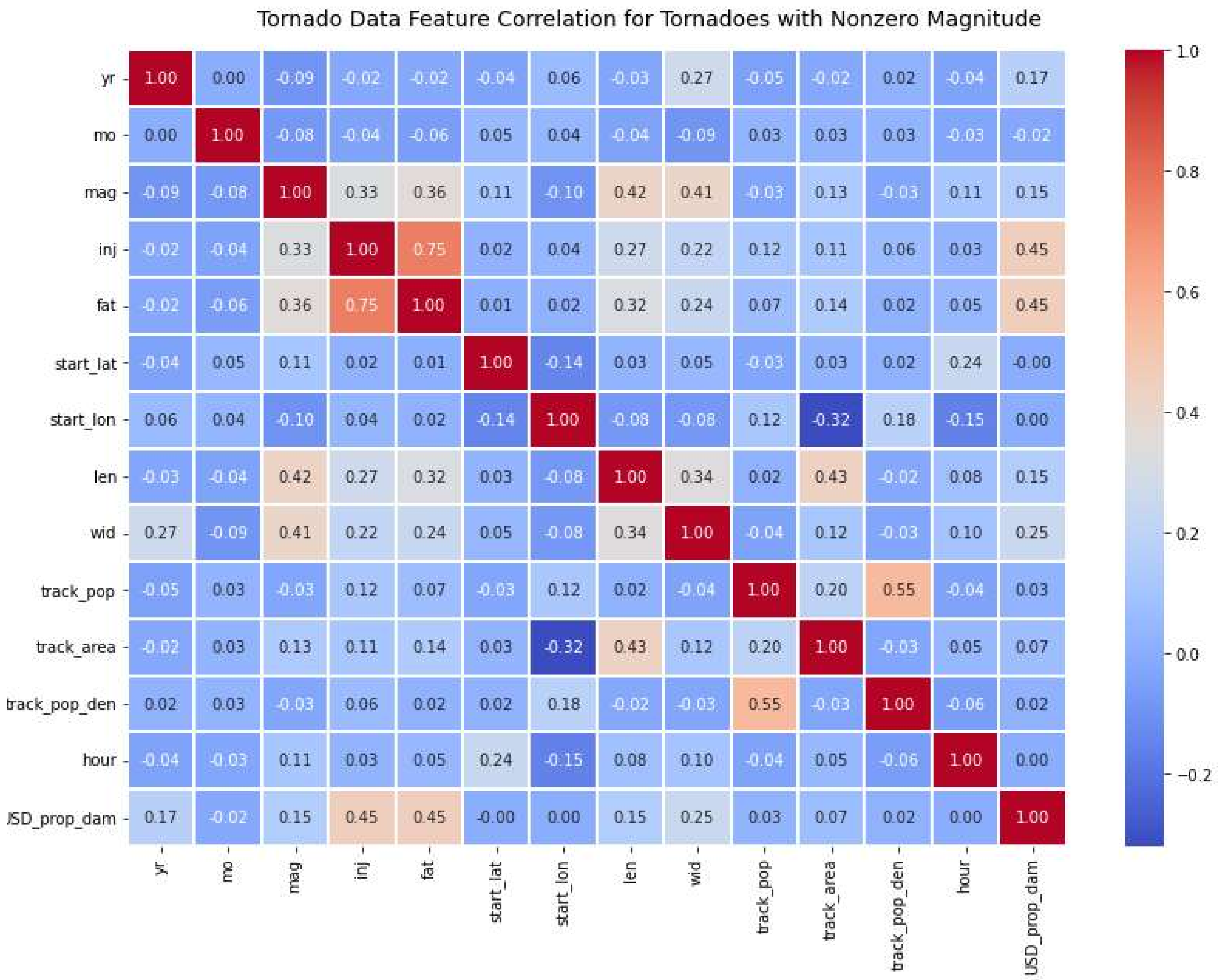
Do tornadoes tend to strike at a certain time of day? It appears from the data that tornadoes are most frequently an afternoon occurrence, becoming most frequent in the hours between 2 and 8 PM.



Does a tornado striking at night have a significant impact on the likeliness of injuries and deaths? The trend here is less clear, but it does seem that tornadoes striking between 10PM and 3AM tend to be slightly more dangerous.



The final step of the EDA process was to produce and inspect a heatmap showing correlation values between the different variables.



Positive Correlations of note: USD Property Damage & Injuries (0.45), USD Property Damage and Fatalities (0.45), Width and USD Property Damage (0.25), Width and Year (0.27), Width and Injuries (0.22), Width and Fatalities (0.24), Track Length and Injuries (0.27), Track Length and Fatalities (0.32), Magnitude and Injuries (0.33), Magnitude and Fatalities (0.36), Start Latitude and Hour of day (0.24)

Negative Correlations of note: Start Longitude and Track Area (-0.32), Start Longitude and Hour of day (-0.15)

Preprocessing

Examining the magnitude column showed that approximately 17K of the total ~36K rows contain a magnitude value of 0. The EF-scale runs from 1-5, so these values were technically incorrect, and couldn't be used as-is for prediction purposes. In the interest of not losing relevant data, however, I chose to isolate instances of 0 magnitude that still have data in the target variables (injuries and fatalities). I changed the magnitude of those tornadoes to the most common value (1, in this case), and dropped the rest.

The final piece that required attention prior to beginning the modeling step was to finally choose a target variable. Injuries and Fatalities both were both indicators of the human toll for a given tornado, but models can't effectively operate with more than one target measure. In the end, I decided that the difference between injuries and deaths in the chaotic environment of a tornado strike often comes down primarily to chance, so I chose to create a new combined metric labeled simply as 'harm', which is a binary variable that takes on a value of 1 for the presence of either/both injuries or deaths, or 0 for neither. It was this new metric that I aimed to predict by employing a collection of Machine Learning algorithms.

Machine Learning Modeling and Analysis

After data-wrangling and preprocessing, I was left with a total of 13 predictor variables:

- yr: The year in which the tornado occurred.
- mo: The month in which the tornado occurred.
- mag: The magnitude of the tornado on the EF-scale. This ranges from 1-5, with 5 being the largest and most destructive.
- len: The length of the tornado's on-ground track, measured in linear miles.
- wid: The width of the tornado funnel, measured in linear yards.
- no_counties: The number of counties through which the tornado traveled.
- track_pop: The total population for all counties in the tornado's track.
- track_area: The total land-area for all counties in the tornado's track.
- track_pop_den: The combined population density for all of the counties in the tornado's track.
- USD_prop_dam: The rough US-dollar value of property damage caused by the tornado. This measure is tiered.
- hour: The hour of day in which the tornado first touched down.
- start_lat: The latitude coordinate (in degrees) for the tornado's origination point.
- start_lon: The longitude coordinate (in degrees) for the tornado's origination point.

And the variable to be predicted:

- harm: An "either/or" combination of Injuries and Deaths. If a tornado caused either or both, it has a harm value of 1, otherwise its value is 0

I then scaled the data using scikit-learn's StandardScaler, which rescales all the data to have a mean of 0 and a standard deviation of 1, thereby preventing variables with larger magnitudes from overpowering the model. After that, all that remained was to split the data into a training and test set. I chose an 80/20 Train/Test split for this dataset.

This was a classification problem with labels, so supervised learning models were the appropriate choice. I deployed and compared results for the following 6 supervised learning models:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbor (KNN)
- Support vector machine (SVM)
- Random Forest
- Gradient Boost

Each of these model types have additional parameters that must be set (known as hyperparameters), and optimal values are not always intuitive. As such I chose to deploy Random Search CV to optimally tune these parameters. Research has shown that Random Search performs nearly as well as the more robust GridSearchCV while being far less resource hungry. Since I had several different models to evaluate, I viewed this as a worthwhile tradeoff.

One additional consideration I had to make was how to compare the performance of the different models. The the number of cases where harm occurred is only a small percentage of the overall number of tornado events (~17% to be precise). That means, for this unbalanced dataset, that a model designed to only choose "no harm" would be "correct" 83% of the time. As such, model accuracy, the most straightforward measure, is not an attractive option in this case.

Instead, I chose to use two other measures: **f1-score** and **Area under the ROC curve**. The f1-score is a weighted average of *precision* and *recall*, where precision is the ratio of True Positives to all positive predictions (TP+FP), and recall is the ratio of True Positives to the sum of True Positives and False Negatives. Recall is also known as sensitivity. f1-scores are useful in that they attempt to strike a balance between minimizing False Positives (in this case predicting harm when none occurred) and minimizing False Negatives (predicting no harm when harm occurred).

The ROC curve is a plot of the True Positive Rate on the y-axis, and the False Positive Rate on the x-axis. Intuitively, this means a "perfect" model that never misclassifies will be a rectangle with points at (0,1) and (1,0). The area under this idealized curve is 1.0, or 100%. As such, we can use the area under our actual ROC curves to evaluate the effectiveness of our models, with AUC (area under curve) values indicating better performance as they increase toward 1.0.

The problem we are modeling is a classification problem with labels, so we will deploy and compare the results for the following supervised learning models:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbor (KNN)
- Support vector machine (SVM)
- Random Forest
- Gradient Boost

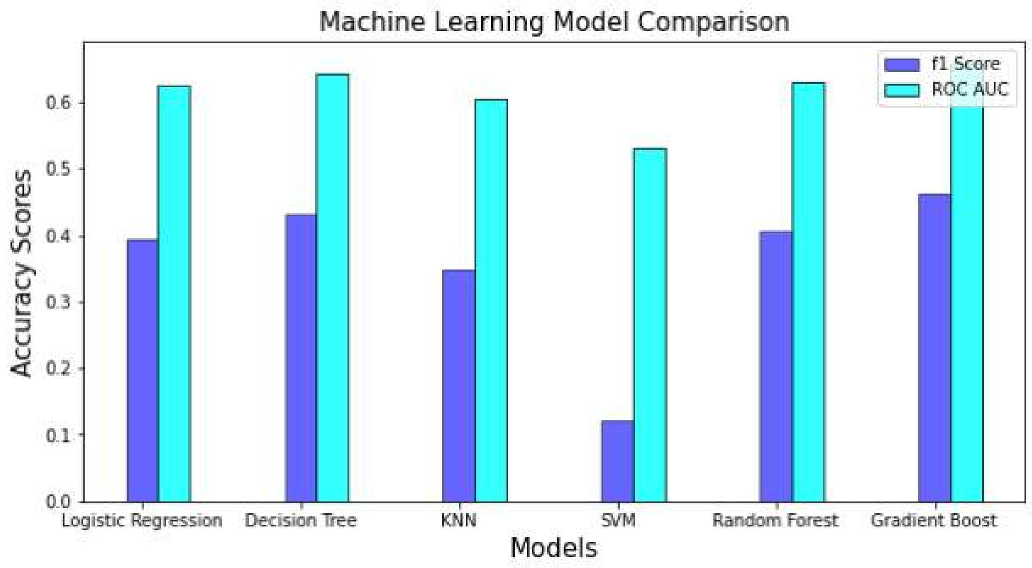
Each of these model types have additional parameters that must be set (known as hyperparameters), and optimal values are not always intuitive. As such we will deploy Random Search CV with 5-fold cross-validation to optimally tune these parameters. Research has shown that Random Search performs nearly as well as the more robust GridSearchCV while being far less resource hungry. Since we have several different models to evaluate, we view this as a worthwhile tradeoff.

Also, a note on how we will compare the models in terms of performance: The dataset we are using, with "harm" as the predicted variable, is very unbalanced. This means that the number of cases where harm occurred is only a small percentage of the overall number of tornado events (~17% to be precise). That means that a model designed to only choose "no harm" would be "correct" 83% of the time. As such, model accuracy, the most straightforward measure, is not an attractive option in this case.

Instead, we'll be using two other measures: **f1-score** and **Area under the ROC curve**. The f1-score is a weighted average of *precision* and *recall*, where precision is the ratio of True Positives to all positive predictions (TP+FP), and recall is the ratio of True Positives to the sum of True Positives and False Negatives. Recall is also known as sensitivity. f1-scores are useful in that they attempt to strike a balance between minimizing False Positives (in this case predicting harm when none occurred) and minimizing False Negatives (predicting no harm when harm occurred).

The ROC curve is a plot of the True Positive Rate on the y-axis, and the False Positive Rate on the x-axis. Intuitively, this means a "perfect" model that never misclassifies will be a rectangle with points at (0,1) and (1,0). The area under this idealized curve is 1.0, or 100%. As such, we can use the area under our actual ROC curves to evaluate the effectiveness of our models, with AUC (area under curve) values indicating better performance as they increase toward 1.0.

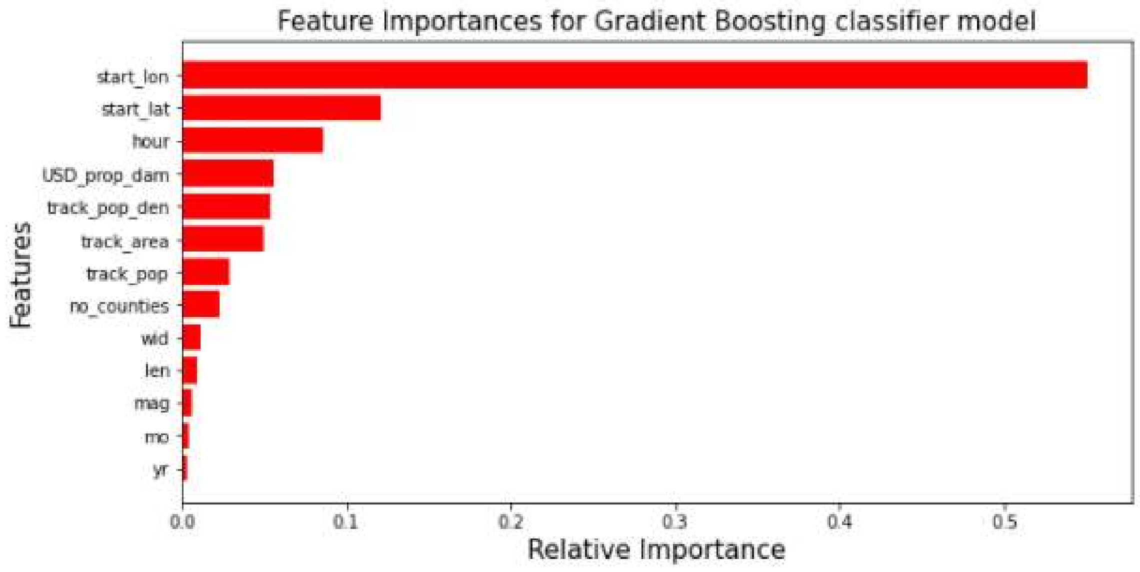
After running all the models on the training and test sets, I compared their performance on the test set using the two measures mentioned above. The results in the chart below show that the Gradient Boost model was the most predictive, followed by Decision Tree and Random Forest. Also of note is how poorly Support Vector Machines performed in this case.



The next step was to focus on the Gradient Boost model and look at which features had the largest impact on predictive power. According to the feature importances chart below, starting longitude is the most predictive feature by a significant margin. This is perhaps an unexpected result, but when one considers that population density tends to increase as you move from the Great Plains toward the East Coast, it seems like a more meaningful outcome. Starting latitude is the second most significant feature, and again, thinking in terms of population density as you move South toward the Gulf Coast, it doesn't appear unreasonable.

Next most important are hour and USD Property Damage. Both make sense, as tornadoes that strike in the middle of the night and tornadoes that cause higher levels of structural damage are both intuitively more likely to cause injuries and deaths.

Perhaps surprising is the fact that tornado width, track length, and magnitude are close to the bottom of the list of feature importances. But, considering that even a comparatively "weak" or "small" tornado still has more than enough power to destroy homes and other buildings, this begins to seem sensible as well.



Conclusion and Next Steps

Given the massive number of variables outside the scope of consideration that can go into determining loss of life in a tornado event, I feel positively about the results achieved by the models, especially the Gradient Boosting model that achieved the highest performance scores.

I'm also fully aware, however, that there is still plenty of room for improvement. In the end, aside from county population density data, we only pressed forward with predictor variables present in the base dataset. As such, there are other variables to consider whose addition could yield an even more robust model.

Some possibilities include:

- Historical climate data by county: I attempted to include climate data to classify the month and year a tornado occurred in as wet/normal/dry and warm/normal/cool relative to historical climate data for its origination county, but in the end, I unable to effectively scale these values in a way that prevented them from being over-weighted in the models. This is an area where a few tweaks could potentially yield significant results.
- Home Age data by county: I was able to find data that reasonably approximated this measure, but again, it didn't prove to be workable in its current form. Still, it would be interesting to see if improvements in building technology have resulted in any decrease in human toll caused by tornadoes.
- Long-term weather patterns (e.g., El Niño/La Niña): Do years under a certain weather pattern result in more or deadlier tornadoes?

- Prevalence of basements in homes: This might be a tricky measure to find, but homes closer to sea level tend to be built without basements, which removes one of the most effective sheltering options in the event of a tornado strike.