Wildfires

A capstone project by:

Marty Hauck, John Osire, Mark Harris, Harold Haugen, Nahid Boustani, and Osie David

Georgetown University Certificate in Data Science

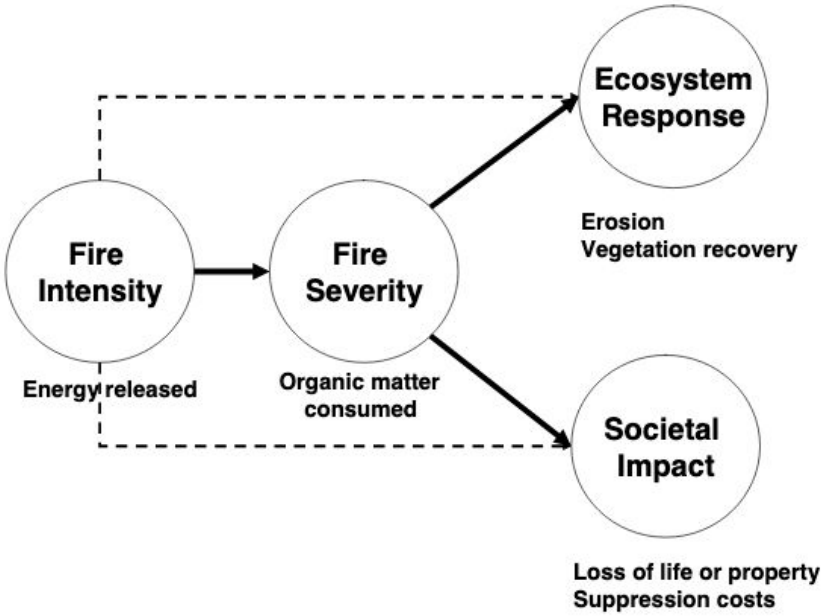
June 20, 2020

Table of Contents

Abstract

Given the concern with climate change affecting the rise of global temperatures, it is plausible that wildfires contribute to the problem of climate change and, inversely, the increase in atmospheric temperatures impacts the severity of wildfires. Wildfires emit carbon dioxide and other greenhouse gases that will continue to warm the planet well into the future and contribute to the ill-effects of global warming. Additionally, smoke from wildfires causes serious health risks that can be deadly for sensitive populations with underlying health conditions. Loss of property and financial distress is another social impact of the wildfires occurring in residential or commercial areas (See diagram below). Hence, our capstone team is exploring a hypothesis that data science modeling can approximate the intensity of wildfires and this will enable national responders to anticipate and react to the danger of these natural events.

Immediate Environmental and Social impacts of Wildfires:

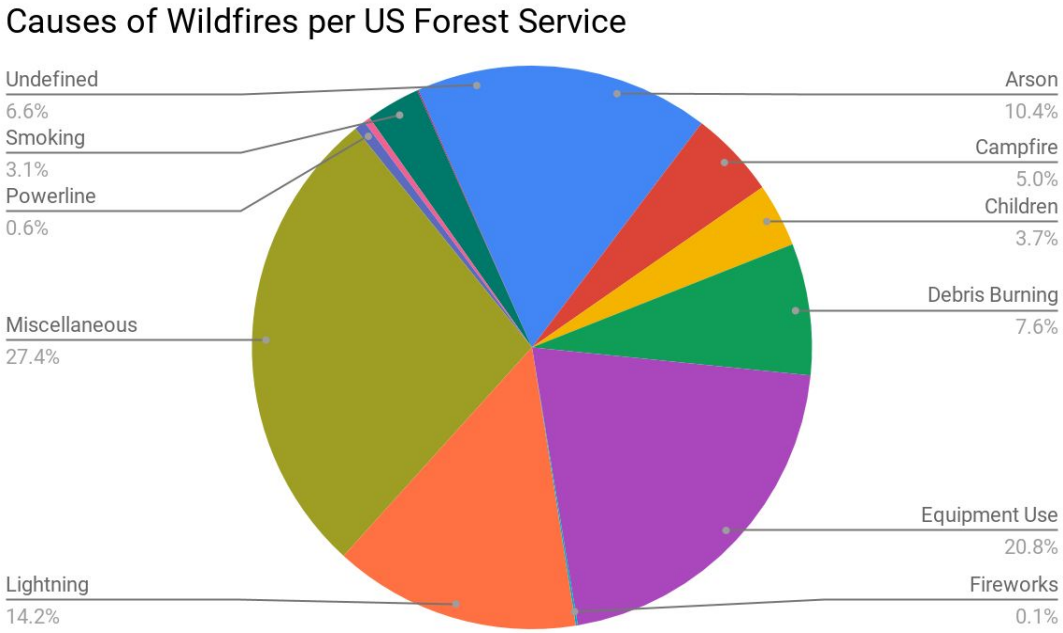


This project was initiated with the intention to address the effect of this natural phenomenon in a novel way and to make a pioneering effort for future endeavors. To do so, we assembled data from three primary sources that affect the intensity of a wildfire and the amount of CO2 emission.

The goal of this capstone is to use machine learning to predict the intensity of wildfires and to see if the five elements comprising the data science pipeline lead to a viable model that speaks to forecasting the power of wildfires: accordingly, is it possible to construct a model that will predict the danger of future wildfires?

Introduction

There are multiple reasons for why wildfires occur and an initial review of data from the United States Department of Agriculture (USDA) between the years of 2003 to 2015 highlights the following causes:



In order to explore the hypothesis on modeling wildfire intensity, the team needed to first understand the data that affects wildfires in terms of what causes a wildfire event and how measurable these features are; these items would comprise our independent fire variables. Then, we would examine the effects of a fire in terms of the byproducts and use these outputs as the dependent fire variables. The team ingested data from government organizations such as the USDA and North American Space Agency (NASA), as well as a commercial entity owned by Apple called Dark Sky. Upon completing wrangling, feature production, and normalization, the team recognized that there were on the order of 20 key features and hundreds of thousands of instances characterizing the data. Through further evaluation of the three data sets, we identified key dependent variables among the features that we thought would best respond to our modeling efforts. Understanding the three key data sets became clearer as we progressed through the five data science pipeline elements of data ingestion, munging and wrangling, computation and analysis, modeling and application, and finally reporting and visualization. Although we applied all of the modeling techniques introduced primarily through the Machine Learning Course, we narrowed down our application to the following estimating techniques:

- Bagging Classifier
- Extra Tree Classifier
- Random Forest Classifier
- GaussianNB
- SVC
- Adaboost Model

To scale, manage, and reduce the amount of data to ensure a quality approach to selecting appropriate machine learning models, we applied binning techniques to group numbers or change the distribution of continuous data to meet our needs.

Hypothesis and Motivation

The original hypothesis for Predicting Wildfires was that every fire event has a unique fire intensity that determines its capacity to spread. At the start of the project we thought that the data features that drove fire intensity were elementary and, that by simply identifying organic matter that burns, we could ascertain the strength of a fire. However, once we started to analyze the data through the data science pipeline, we recognized there are multiple variables that characterize a fire and that our hypothesis needed to evolve through understanding more than one data feature. In terms of dependent factors influencing a wildfire, we recognized potential 'y' variables that would help with developing a predictive model as being ECO2 (Emissions Carbon Dioxide) and fire Brightness; the latter term being equivalent to the fire intensity (temperature of the fire pixel measured in Kelvin) as measured by a NASA satellite. Both of these variables are continuous which enabled us to binnerize them in order to construct a classification model.

Relative to specific fire events, we learned that by combining fire attributes with local weather data we could predict the ECO2 emissions and the fire brightness on a scale equivalent to low, med, and high categories. We also came to recognize secondary project objectives pertaining to the element of time and how this affects our model's ability to predict the duration of a fire and burn size. We further learned that we could adjust the independent variables we collected by tweaking them based on our feature analysis. As the project progressed through exploratory data analysis, we observed clear correlations between the burning of deciduous vegetative matter on the forest floor called 'cover type', dead trees that are referred to as 'coarse woody debris', and trees that are alive. Although we recognized that CO2 emissions clearly correlated with the burning of organic matter, we chose to focus on Brightness as the feature that would best align with wildfire intensity. The end result was that predicting wildfire intensity was a process of aggregating a multitude of data instances, synthesizing the data through iteration, and applying machine learning approaches to extracting a meaningful representation of what a wildfire prediction model could look like.

The team's motivation for selecting a project effort such as predicting wildfire intensity was grounded in a shared motivation to understand the effects of climate change on our world and that wildfire events are singularly devastating to the environment and significantly contribute to greenhouse gas emissions. As citizens of the world, wildfire events can impact almost any country and depending on the unique aspects of a nation's geography and environment, wildfire models can enhance a state's ability to anticipate and respond to a fire event.

Data Sources

The team focused on three primary sources of data drawn from nationally recognized sources associated with provenance and veracity from the 2003 through 2015. These included:

A. Emissions - Data gathered from the USDA provided us metrics on emissions, a single burn severity category, and other related geographic details such as covertype (e.g., the type of woodland area related to a specific geospatial location). Metrics in total amounted to 22 columnar features, and 7.2m rows we could draw from; most of which were integer based, but some were qualitative in nature with descriptions amounting to name, location, and type of data.

a. Short bio from source site - The Missoula Fire Lab Emission Inventory (MFLEI) is a retrospective, daily wildfire emission inventory for the contiguous United States with a spatial resolution of 250 meters (m). **See Table 1, 2 and 3.**

B. Intensity - NASA's Fire Information for Resource Management System (FIRMS) provides Near Real Time (NRT) fire data from NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) and Visible Infrared Imaging Radiometer Suite (VIIRS) satellites. **See Table 4**. We acquired data from NASA that perhaps furnished us with the singularly most important attribute to assess the magnitude of a fire: Three data metrics that represented the intensity of fire activity per view of two separate satellite recordings with 2.15m records across latitude and longitude coordinates. These included:

a. Brightness - Brightness temperature 21 (Kelvin)

b. Bright_T31 - Brightness temperature 31 (Kelvin), and

c. FRP - Fire Radiative Power (MW - megawatts)

C. Weather - Finally, as a foundational component of our hypothesis, we looked to source weather data from Apple's DarkSky API to correlate weather features across the two datasets (USDA and NASA datasets) in order to form a model ready set of X features to support our perceived model. **See Table 5**

Table 1 - Cover Type Feature

Covertype 0 =	Unclassified/Open Water/ Perennial Ice/Snow/Barren Land;
1 =	herbaceous;
2 =	shrub/scrub;
3 =	forest;
21-24 =	developed;

81 =	hay/pasture;
82 =	cultivated crop;
90 =	woody wetlands;
95 =	emergent herbaceous wetlands

Table 2 - Fuel Code Feature

Fuel Code Category		
0 Non-fuel	1 Herbaceous	2 Shrub / scrub
1100 White / red / jack pine group	1120 Spruce / fir group	1140 Longleaf / slash pine group
1160 Loblolly / shortleaf pine group	1180 /2180 Pinyon / juniper group	1220 Ponderosa pine group
1240 Western white pine group	1260 Fir / spruce / mountain hemlock group	1280 Lodgepole pine group
1300 Hemlock / Sitka spruce group	1320 Western larch group	1340 Redwood group
1360 Other western softwoods group	1370 California mixed conifer group	1380 Exotic softwoods group
1400 Oak / pine group	1500 Oak / hickory group	1600 Oak / gum / cypress group
1700/2700 Elm / ash / cottonwood group	1800 Maple / beech / birch group	1900/2900 Aspen / birch group
1910 Alder / maple group	1920 Western oak group	1940 Tanoak / laurel group
1950/2950 Other western hardwoods group	1980 Tropical hardwoods group	1990 Exotic hardwoods group

Table 3 - Fuel Moisture Class Code Features

Fuel Moisture Class Categories			
1 = very dry	2 = dry;	3 = moderate;	4 = moist

Table 4 - Satellite Fire Attributes

Attribute	Short Description	Long Description
Latitude	Latitude	Center of 1km fire pixel but not necessarily the actual location of the fire as one or more fires can be detected within the 1km pixel.
Longitude	Longitude	Center of 1km fire pixel but not necessarily the actual location of the fire as one or more fires can be detected within the 1km pixel.
Brightness	Brightness temperature 21 (Kelvin)	Channel 21/22 brightness temperature of the fire pixel measured in Kelvin.
Scan	Along Scan pixel size	The algorithm produces 1km fire pixels but MODIS pixels get bigger toward the edge of scan. Scan and track reflect actual pixel size.
Track	Along Track pixel size	The algorithm produces 1km fire pixels but MODIS pixels get bigger toward the edge of scan. Scan and track reflect actual pixel size.
Acq_Date	Acquisition Date	Data of MODIS acquisition.
Acq_Time	Acquisition Time	Time of acquisition/overpass of the satellite (in UTC).
Satellite	Satellite	A = Aqua and T = Terra.
Confidence	Confidence (0-100%)	This value is based on a collection of intermediate algorithm quantities used in the detection process. It is intended to help users gauge the quality of individual hotspot/fire pixels. Confidence estimates range between 0 and 100% and are assigned one of the three fire classes (low-confidence fire, nominal-confidence fire, or high-confidence fire).
Version	Version (Collection and source)	Version identifies the collection (e.g. MODIS Collection 6) and source of data processing: Near Real-Time (NRT suffix added to collection) or Standard Processing (collection only). "6.0NRT" - Collection 6 NRT processing. "6.0" - Collection 6 Standard processing. Find out more on collections and on the differences between FIRMS data sourced from LANCE FIRMS and University of Maryland.
Bright_T31	Brightness temperature 31 (Kelvin)	Channel 31 brightness temperature of the fire pixel measured in Kelvin.
FRP	Fire Radiative Power (MW - megawatts)	Depicts the pixel-integrated fire radiative power in MW (megawatts).
Type*	Inferred hot spot type	0 = presumed vegetation fire 1 = active volcano 2 = other static land source 3 = offshore
DayNight	Day or Night	D= Daytime fire, N= Nighttime fire

Table 5 - Weather Information

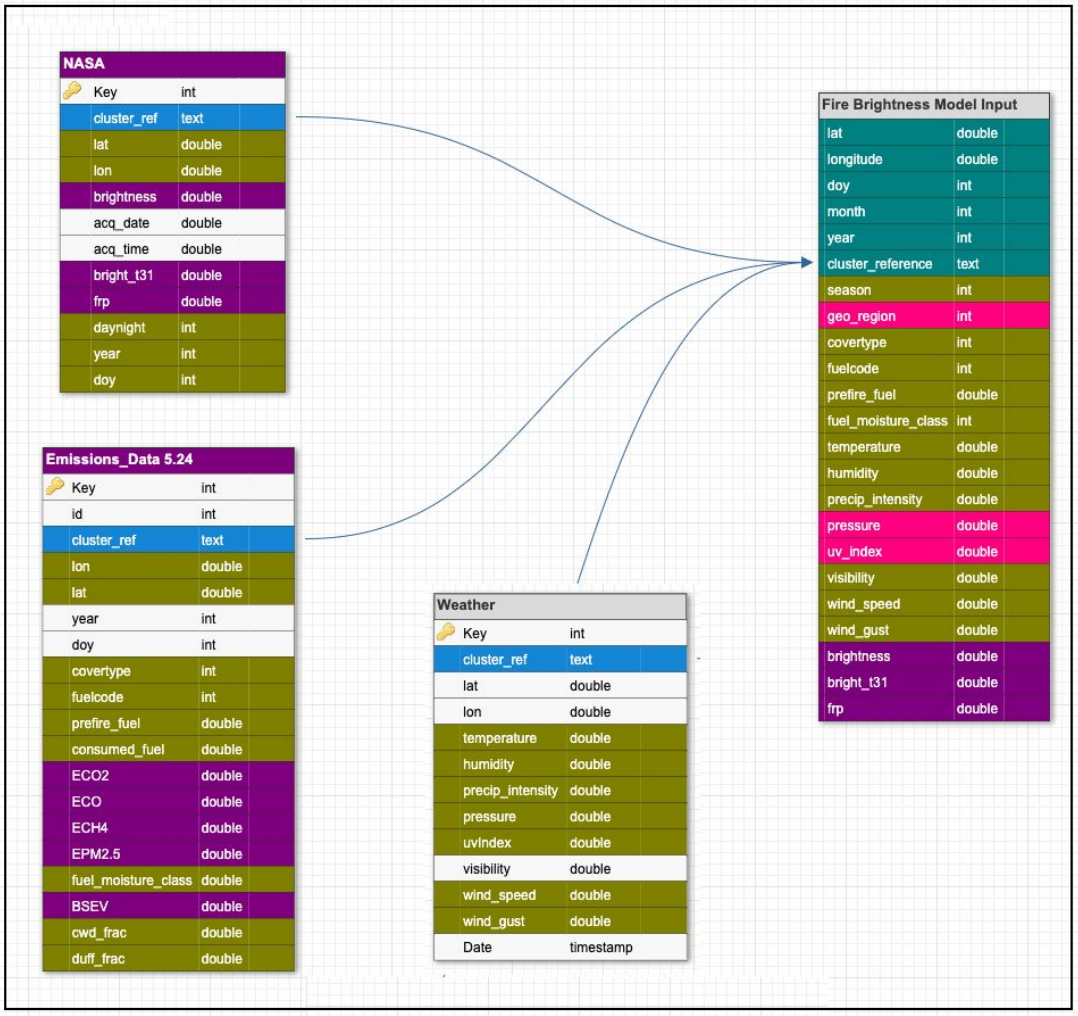
Weather Field	Description
apparentTemperature	The apparent (or “feels like”) temperature in degrees Fahrenheit.
cloudCover	The percentage of sky occluded by clouds, between 0 and 1, inclusive.
dewPoint	The dew point in degrees Fahrenheit.
humidity	The relative humidity, between 0 and 1, inclusive.
icon	A machine-readable text summary of this data point, suitable for selecting an icon for display. If defined, this property will have one of the following values: clear-day, clear-night, rain, snow, sleet, wind, fog, cloudy, partly-cloudy-day, or partly-cloudy-night. (Developers should ensure that a sensible default is defined, as additional values, such as hail, thunderstorm, or tornado, may be defined in the

	future.)
precipAccumulation	The amount of snowfall accumulation expected to occur (over the hour or day, respectively), in inches. (If no snowfall is expected, this property will not be defined.)
precipIntensity	The intensity (in inches of liquid water per hour) of precipitation occurring at the given time. This value is conditional on probability (that is, assuming any precipitation occurs at all).
precipProbability	The probability of precipitation occurring, between 0 and 1, inclusive.
precipType	The type of precipitation occurring at the given time. If defined, this property will have one of the following values: "rain", "snow", or "sleet" (which refers to each of freezing rain, ice pellets, and “wintery mix”). (If precipIntensity is zero, then this property will not be defined. Additionally, due to the lack of data in our sources, historical precipType information is usually estimated, rather than observed.)
pressure	The sea-level air pressure in millibars.
summary	A human-readable text summary of this data point. (This property has millions of possible values, so don’t use it for automated purposes: use the icon property, instead!)
temperature	The air temperature in degrees Fahrenheit.
time	The UNIX time at which this data point begins. minutely data point are always aligned to the top of the minute, hourly data point objects to the top of the hour, daily data point objects to midnight of the day, and currently data point object to the point of time provided all according to the local time zone.
uvIndex	The UV index.
visibility	The average visibility in miles, capped at 10 miles.
windBearing	The direction that the wind is coming from in degrees, with true north at 0° and progressing clockwise. (If windSpeed is zero, then this value will not be defined.)
windGust	The wind gust speed in miles per hour.
windSpeed	The wind speed in miles per hour.

Data Storage

We chose to use MySQL Relational Database Management System (RDMS) for storing our data because of the need to provide a central location as our team would be accessing this data on network server concurrently and on demand. MySQL database offers features that can allow many users to read/write at the same time. Much less complex and easier to use once set up and provides consistency (no data corruption). It provides flexibility and can have different tables for different models, relations between data and a query language to effectively retrieve this data. In terms of data types that we needed for our fire data, MySQL has more data types other than just numeric or string. In terms of strings data types, we have CHAR and VARCHAR character types. MySQL also provides us with TEXT type that has more features which CHAR and VARCHAR cannot cover. TEXT is useful for storing long-form text strings. Numeric: DECIMAL, FLOAT, DOUBLE, TINYINT, SMALLINT, MEDIUMINT and INT. Date and Time Types: DATE, TIME, DATETIME, TIMESTAMP and YEAR. MySQL offers other data types but will not be included in this paper because we are using a limited number of data types needed for this project. Databases can handle complex queries, as opposed to using csv files that involve a lot of manual processes.

The diagram below shows our database model:



Data Ingestion

While many of our sources of data came from CSV output and we were able to easily load these into the database, weather data was being pulled using the DarkSky API. In order to tap into this we used the darksky library to connect into the api and created a function that would loop through the rows of a dataframe and call each row to the DarkSky API to then store them into a dictionary. Once pulled into a dictionary we then stored them into a dataframe and did some light manipulation by removing columns that were inconsistently pulled. After the dataframe was clean we would use sqlalchemy to push the new weather data-points into our loading database. This can be seen in the DarkSkyWeatherPull notebook.

Munging / Wrangling

Several techniques were called upon to pull our data sources/ information together into a digestible format. These techniques included:

1. **Combining** - the Emissions datasets were downloaded as individual yearly CSV files from the USDA website for years including Each file then needed to be combined to form a consolidated dataset for input into the database.

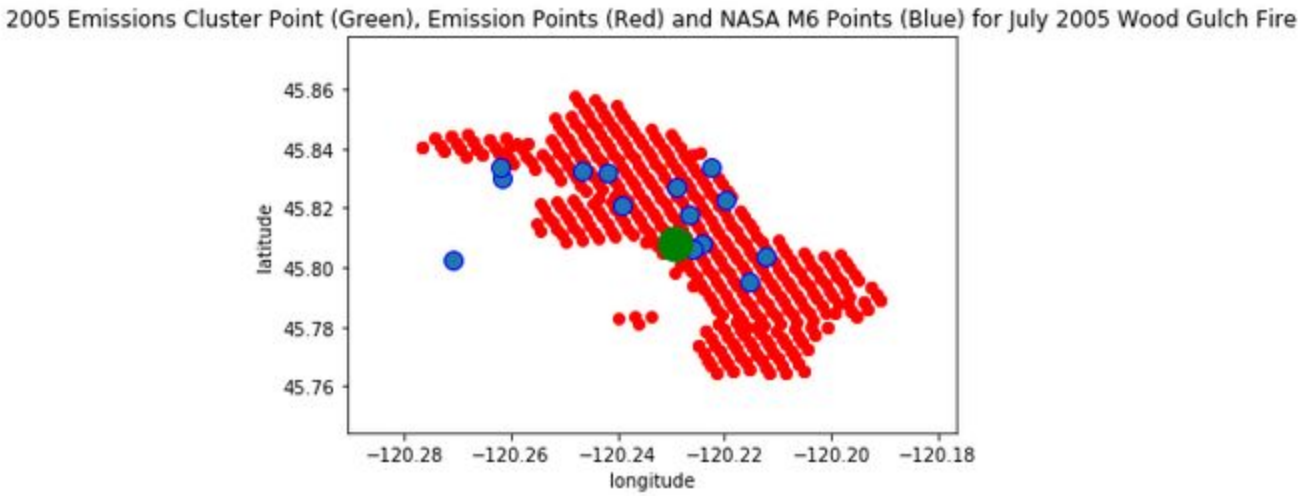
2. **DBScan / Reduction of geospatial coordinates into manageable clusters** - A key issue for the team was use of the Darksky API and the ability to pull weather data for such a large number individual records between the Emissions (7.25m) and NASA M6 (2.15m) records. Our team turned to DBScan as a method to reduce the geographic points into more manageable clusters that would allow for the team to:

    a. reduce the number of API calls by being able to pull weather data for smaller geographic regions that would essentially experience the same weather conditions,

    b. define an identification and labeling scheme to connect information between the Emissions and NASA M6 data sets,

    c. consolidate the individual feature data into truly unique fire events that the team could analyze further from an individual fire perspective that would include fire size, event discovery and end dates, area growth during the event time period and other labeling information such as fire name and additional descriptive information. See Figure 17 for representation of a unique fire event with Emissions and NASA M6 information plotted around the determined centerpoint.

Figure 17 - (Blue represents NASA M6 fire readings, Red Emission readings, Green centerpoint for selected cluster reference)



DBSCan was the team's first use of a non-supervised machine learning algorithm very early in the capstone project to define and label unique fire events. Our team determined unique fire clusters for the years between 2003 and 2015, capturing the centerpoints for each cluster, related Emissions based information for each centerpoint, as well as tagging each underlying lat/long location from the Emissions dataset with a corresponding centerpoint reference label to be used later to merge information across datasets.

We utilized the following code to support our clustering:

Key views of DBScan Code:

```
### Running DBScan and setting parameters including Min Sample, Algorithm, Metric.
start_time = time.time()
db = DBSCAN(eps=epsilon, min_samples=1, algorithm='ball_tree', metric='haversine').fit(np.radians(coords))
cluster_labels = db.labels_

# Determine the number of clusters
num_clusters = len(set(cluster_labels))

# Print the outcome
message = 'Clustered {:,} points down to {:,} clusters, for {:.1f}% compression in {:,.2f} seconds'
print(message.format(len(emyearset20xx), num_clusters, 100*(1 - float(num_clusters) / len(emyearset20xx)), time.time()-start_time))
print('Silhouette coefficient: {:0.03f}'.format(metrics.silhouette_score(coords, cluster_labels)))
```

Capturing individual data points (lat/long) that make up clusters into DataFrame

```
### Determining the number of clusters developed from DBScan and the cluster point count in the first cluster.
ab = list(zip(clusters[0]))
ab = len(ab)
print('Number (len) of Clusters from results = ', len(clusters))
print('Number of cluster datapoints in the first Cluster = ', ab)
```

```
Number (len) of Clusters from results =  8234
Number of cluster datapoints in the first Cluster =  359
```

```
#clusters[0]
```

```
### Loop to pull in the cluster points that make up each Cluster represented in the Series 'Clusters'.
def preparecluster(row, clusterseries):
    clusterDF = pd.DataFrame(clusterseries[0])
    clusterDF = clusterDF.assign(ClusterNum=row)
    row = 1
    while row < len(clusterseries):
        clusterdata = (clusterseries[row])
        clusterDFTemp = pd.DataFrame(clusterdata)
        clusterDFTemp = clusterDFTemp.assign(ClusterNum=row)
        clusterDF = clusterDF.append(clusterDFTemp, ignore_index=True)
        row = row + 1
    return clusterDF
```

```
### Viewing results of dataframe consolidation and adding year to
groupedclusters = preparecluster(0,clusters)
groupedclusters = groupedclusters.assign(Year=emyear)
groupedclusters.tail()
```

	0	1	ClusterNum	Year
375836	48.9261	-122.1427	8233	2009
375837	48.9277	-122.1468	8233	2009
375838	48.9283	-122.1436	8233	2009
375839	48.9289	-122.1404	8233	2009
375840	48.9305	-122.1445	8233	2009

```
groupedclusters.to_csv('../data/Emclustermakeup_2009.csv', encoding='utf-8')
```

Find the point in each cluster that is closest to its centroid

DBSCAN clusters may be non-convex. This technique just returns one representative point from each cluster. First get the lat,lon coordinates of the cluster's centroid (shapely represents the first coordinate in the tuple as x and the second as y, so lat is x and lon is y here). Then find the member of the cluster with the smallest great circle distance to the centroid.

```
def get_centermost_point(cluster):
    centroid = (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
    centermost_point = min(cluster, key=lambda point: great_circle(point, centroid).m)
    return tuple(centermost_point)

centermost_points = clusters.map(get_centermost_point)
```

Results - Through the DBScan process, we clustered 5.96m unique lat/longs within the Emissions dataset into 109,312 clusters. A separate process was performed to create a unique cluster reference label based on the centerpoint year and cluster label assigned during clustering, to then assign the new cluster reference label to the individual data points within the Emissions and eventually NASA M6 and Weather datasets. **See Appendix 1-A** for our results of the clustering process by (number of records, number of clusters, compression %, and a derived Silhouette Score (between -1 and 1)**. We documented future considerations and hurdles with the Next Steps section below.**

K-D Tree to Identify and Assign Cluster Reference to Closest NASA M6 Feature - Upon establishing the cluster areas, we were in need to assign a cluster reference that was generated for the centerpoints to the underlying and individual latitude/longitude points making up each cluster across the Emission dataset. These cluster references were then assigned to the NASA M6 information using Year, Latitude and Longitude in order to facilitate movement of feature information between the two primary datasets. The Lat/Long for each centerpoint were also used in the pulling of weather information by cluster geographic area from the Weather API.

We set up a K-D Tree algorithm to transform the latitude and longitude coordinates across the cluster centerpoint data by year, to then determine the closest coordinate distance between NASA M6 latitude and longitude and the centerpoints, then the development of a dataframe that capture the Cluster Reference, Lat/Long and Year from the Centerpoint data.

Setting Up K-D Tree Lat/Long Inputs and Year

```
# Set year for centerpoint data to support target search for closest NASA M6 location to cluster location.
centerpoints_xx = centerpoints_0305[(centerpoints_0305.year == 2008)]
```

```
# Set year for the NASA M6 data.
NASA_M6_xx = NASA_M6[(NASA_M6.year == 2008)]
print(NASA_M6_xx.shape)
```

```
(119797, 19)
```

```
# Turning latitude, longitude, doy into cartesian coordinates.
def cartesian(latitude, longitude, doy):
    # Convert to radians
    latitude = latitude * (math.pi / 180)
    longitude = longitude * (math.pi / 180)

    R = 6371 # 6378137.0 + elevation  # relative to centre of the earth
    X = R * math.cos(latitude) * math.cos(longitude)
    Y = R * math.cos(latitude) * math.sin(longitude)
    #Z = R * math.sin(latitude)
    Z = 2 * doy
    return (X, Y, Z)
```

```
# Placing latitude/longitude from target dataframe into a list.
centerpoint_places = []
for index, row in centerpoints_xx.iterrows():
    coordinates = [row['latitude'], row['longitude'], row['doy']]
    cartesian_coord = cartesian(*coordinates)
    centerpoint_places.append(cartesian_coord)

tree = spatial.KDTree(centerpoint_places)
centerpoint_places[1]
```

```
(907.4565604008313, -5681.440434218411, 640.0)
```

```
# Set function for K-D Tree to search list of coordinates in target dataframe for closest match.
def find_centerpoint(lat, lon, doy):
    cartesian_coord = cartesian(lat, lon, doy)
    closest = tree.query([cartesian_coord], p = 2)
    index = closest[1][0]
    return closest
```

Function to assign Cluster Reference label to NASA M6 data.

Functin will determine the closest cluster centerponit and cluster label from the DBScan results
and pull over the Cluster_Reference label to each NASA M6 data record.

```
# Function to iterate through each NASA M6 recrod, identify the closest coordinate match between the NASA M6 record
# and the created cluster centerpoints, then create a new dataframe to capture the distance, result row, lat/long,
# year, and finally cluster reference from the centerpoint to the NASA M6 record.

def find_cluster_ref(sourcedf, targetdf):
    NASApoint_clusterref = pd.DataFrame()
    nasa_index = 0
    nasa_index = int(nasa_index)
    while nasa_index < len(sourcedf):
        #capture data from target/cluster to then use to find target match in other dataframe:
        source_doy = sourcedf.iloc[nasa_index]['doy']
        source_year = sourcedf.iloc[nasa_index]['year']
        source_lat = sourcedf.iloc[nasa_index]['latitude']
        source_long = sourcedf.iloc[nasa_index]['longitude']

        # Running cluster find function:
        distance_location = find_centerpoint(source_lat, source_long, source_doy)
        targetlocation = distance_location[1]
        targetlocation = int(targetlocation)

        # Lines to pull data from the target dataframe, will need to be customized to the target DF.
        target_lat = targetdf.iloc[targetlocation]['latitude']
        target_long = targetdf.iloc[targetlocation]['longitude']
        target_doy = targetdf.iloc[targetlocation]['doy']
        target_clusterref = targetdf.iloc[targetlocation]['cluster_reference']
        target_year = targetdf.iloc[targetlocation]['year']
        #target_discdoy = targetdf.iloc[targetlocation]['DISCOVERY_DOY']
        #target_contdoy = targetdf.iloc[targetlocation]['CONT_DOY']

        # Create new DF pulling in features from Cluster Points file and Target File:
        cdftemp = pd.DataFrame({'source_lat':[source_lat], 'source_long': [source_long], 'source_year':[source_year],
                                'source_doy':[source_doy],'distance': distance_location[0], 'resultrow': distance_location[1],
                                'targetlat':[target_lat], 'targetlong':[target_long],'target_doy':[target_doy],
                                'target_year':[target_year], 'target_clusterref':[target_clusterref]})

        NASApoint_clusterref = NASApoint_clusterref.append(cdftemp, ignore_index = True)
        nasa_index = nasa_index + 1
    return NASApoint_clusterref
```

```
nasa_center_match = find_cluster_ref(NASA_M6_xx,centerpoints_xx)
nasa_center_match.tail(8)
```

	source_lat	source_long	source_year	source_doy	distance	resultrow	targetlat	targetlong	target_doy	target_year	target_clusterref
119789	32.3796	-89.2364	2008	366	58.698376	4372	31.6486	-88.8592	357.0	2008	2008_4373
119790	32.3792	-89.2441	2008	366	59.123724	4372	31.6486	-88.8592	357.0	2008	2008_4373
119791	32.3772	-89.2501	2008	366	59.386680	4372	31.6486	-88.8592	357.0	2008	2008_4373
119792	32.7128	-88.6323	2008	366	28.725507	4925	32.2904	-88.4860	366.0	2008	2008_4926
119793	32.7150	-88.6197	2008	366	28.298676	4925	32.2904	-88.4860	366.0	2008	2008_4926

Results - Using the K-D Tree based algorithm, we prepared and combined 13 individual annual files (e.g, 2003 - 2015) into a single dataset of NASA M6 records represented by lat/long, doy and year, with the added distance datapoint (i.e., distance from the NASA M6 fire point record to the cluster centerpoint), reference row and cluster reference labels. This resulting dataframe contained 1.65m records across the years under review. As we moved to transition the cluster reference labels from the K-D Tree results to the base NASA M6 records, we made the decision and assumption to only select those records having a distance of under 150 euclidean distance (or 2.6km) in order to ensure a) the NASA M6 lat/long record and b) weather data pulled for the cluster would be accurate across all of the individual data points within the label area. After applying this assumption, total NASA M6 usable records were reduced from 1.65m to 1.35m records.

3. **Merging / DeDuplication -** Because our team was required to merge feature data from three primary data sources (Emissions, NASA M6, and Weather), several notebooks were tasked with merging information across data frames and then de-duplicating records that appeared due to the merge process. We primarily used a left merge between data frames in the following instances:

    a. **Cluster Reference Label assignment -** we created the cluster reference label by combining the Year and Cluster Label for all centerpoint records coming out of the DBSCan results. The cluster reference label was assigned/merged between:

        i. the 109k cluster centerpoint dataframe to the 5.96m underlying individual cluster points from the DBSCan run in order to create a completed series of all lat/longs with the related cluster reference label.

        ii. the individual cluster points dataframe (5.96m) back to the original Emissions dataset using lat/long and year. Upon de-duplication, this reduced the data set to 5.93m records.

        iii. assigning the cluster reference to the NASA M6 records (2003 to 2015 = via K-D Tree using lat/long, year, and day of year.

    b. **Emissions Features to the Fire Brightness Model -** in order to move feature data from the Emissions Data set, we relied on the cluster reference label as the key to move information between two similar geographic locations.

    c. **Weather Data to the Fire Brightness Model** - weather information that was derived from the DarkSky API using the historical date and cluster centerpoint latitude and longitude was ultimately merged into together with the 1.35m NASA M6 records as we develop the Fire Brightness model input dataframe.

4. **Removing Null Values** -

    a. **Emissions** - Based on analysis of the Emissions dataset, we noted that certain records had null values that spanned across many of the key features that we were intending to use in our model inputs. We removed the null values reducing the file from 7.25m to 5.96m records.

    b. Weather - Depending on the time and latitude and longitude of the api call to DarkSky there were some fields that had a high % of missing values. Based on which fields contained the missing values we chose to drop the field all together (pressure, uv_index) if the missing values exceeded 50% or if the data field was something we felt we couldn't fill in, like the Summary field. The one field that we did augment was the precipAccumulation. We made the assumption that if this was null then there was no rain. In summary, although certain

records had null values, the impact was only to 1.7% of the record population, reducing the total records by 23,499 to 1.32m.

5. **Feature Creation**

   a. **Fire Regions** - With the use of such a large set of geographical coordinates, we chose to use the KMeans algorithm as a method to cluster the Fire Intensity records into a smaller number of geographic regions rather than using each latitude and longitude coordinate as a feature, in order to simplify the feature set. Based on review of wildfire geographic maps and our review of the Elbow Curve analysis, we felt comfortable with setting N for KMeans at 8, although understanding that 4 to 5 may have provided similar results.

Figure 18 - Elbow Curve for Fire Brightness Model Input (see dataframe information below)

```
K_clusters = range(1,15)

kmeans = [KMeans(n_clusters=i) for i in K_clusters]

Y_axis = Fire_Brightness_Model_TestInput[['latitude']]
X_axis = Fire_Brightness_Model_TestInput[['longitude']]

score = [kmeans[i].fit(Y_axis).score(Y_axis) for i in range(len(kmeans))]

# Visualize
plt.plot(K_clusters, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

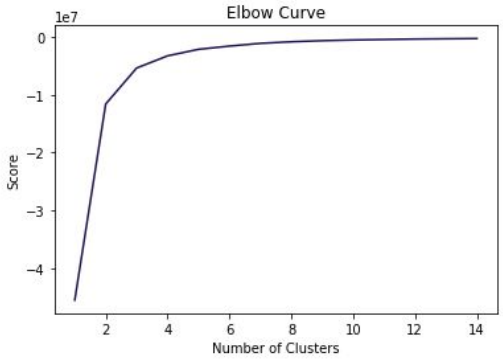
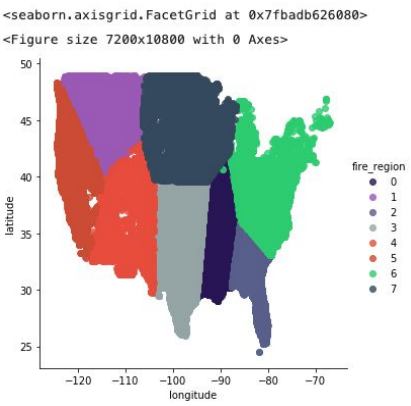


Figure 19 - (KMeans results from performing a clustering process using N=8 regions)



b. **DOY to Four Seasons** - As an additional feature, we decided to reduce the day of year into the four traditional seasons and applied this label as a new feature in the Fire Intensity model input.

6. **Fire Brightness Model Input dataframe -** Lastly in order to prepare a standardized model input dataframe that the team could easily pull down from our SQL database and quickly alter to identify the X and y values, we performed the following data movements:

   a. Used the NASA M6 data with assigned cluster reference labels as the base data set.

   b. Using the cluster reference label, moved feature information ['covertype', 'fuelcode', 'prefire_fuel', 'fuel_moisture_class'] from the emissions dataset to the model input.

   c. Again, using the cluster reference label, moved weather feature information from the weather dataset including [‘temperature', 'humidity', 'precip_intensity', 'visibility' 'wind_speed', 'wind_gust', 'brightness'].

   d. Created and merged the Season and Fire Region labels into the dataframe.

   e. Multiple Y categories - with the NASA M6 dataset having three fire intensity numerical target values present in the data, we chose the ‘Brightness’ satellite reading as our core datapoint. Because we had chosen to pursue a Classification model solution for our capstone, we performed a binning analysis (see Statistical Analysis below) over the distribution to determine category options. As a method to test various Y counts through the test models, we developed four categorical options as follows:

      i. Four equal bins using the basic quartile breakdown for Fire Intensity [Low, Medium, High Severe]

      ii. A Two, Three and Four category based on balanced binning.

Figure 20 - (Fire Intensity Model Input)

```
print(Fire_Brightness_Modelv10.shape)
Fire_Brightness_Modelv10.head()
```

```
(1328922, 23)
```

	latitude	longitude	doy	month	year	cluster_reference	fire_region	season	covertype	fuelcode	prefire_fuel	fuel_moisture_class	temperature
0	34.5954	-78.6218	1	1	2003	2003_4279	6	3	3	1600	6220.097576	3	64.14
1	33.4182	-110.8618	1	1	2003	2003_1522	4	3	3	1220	4534.187262	2	32.17
2	29.7120	-95.1284	1	1	2003	2003_919	3	3	1	1	277.412850	2	65.97
3	28.9161	-98.6293	1	1	2003	2003_777	3	3	1	1	251.296812	2	72.89
4	32.7772	-95.0444	1	1	2003	2003_3100	3	3	1	1	173.172870	2	58.39

humidity	precip_intensity	visibility	wind_speed	wind_gust	brightness	fire_intensity	fire_intensity_twocat	bright_t31	frp
0.88	0.011	9.022000	6.42	12.51	306.5	Low	Moderate	289.2	11.0
0.37	0.000	9.216293	6.88	18.50	307.6	Low	Moderate	285.1	10.8
0.50	0.000	9.997000	10.98	16.90	307.2	Low	Moderate	294.1	5.6
0.51	0.000	9.997000	25.06	28.59	313.3	Medium	Moderate	297.4	12.0
0.30	0.000	9.216293	7.99	10.99	301.3	Low	Moderate	289.9	4.2

Exploratory and Statistical Data Analysis

The project's Exploratory and Statistical Data Analysis (EDA) chapter capitalized on the initial data wrangling and munging process to identify and label features that were important to future machine learning modeling efforts. The team's EDA efforts were divided into two main categories: 1) analyzing the NASA key intensity attributes and pairing these with key features from the USDA dataset, 2) analyzing a unique fire intensity data set (similar to the NASA file) but differentiated with cluster reference labels and quartile breakdowns for Fire Intensity denoted by Low, Medium, High, and Severe categories. With this goal in-mind, multiple rounds of analysis was performed as the team revisited the individual emissions, NASA, and weather datasets as part of the spiral software engineering process.

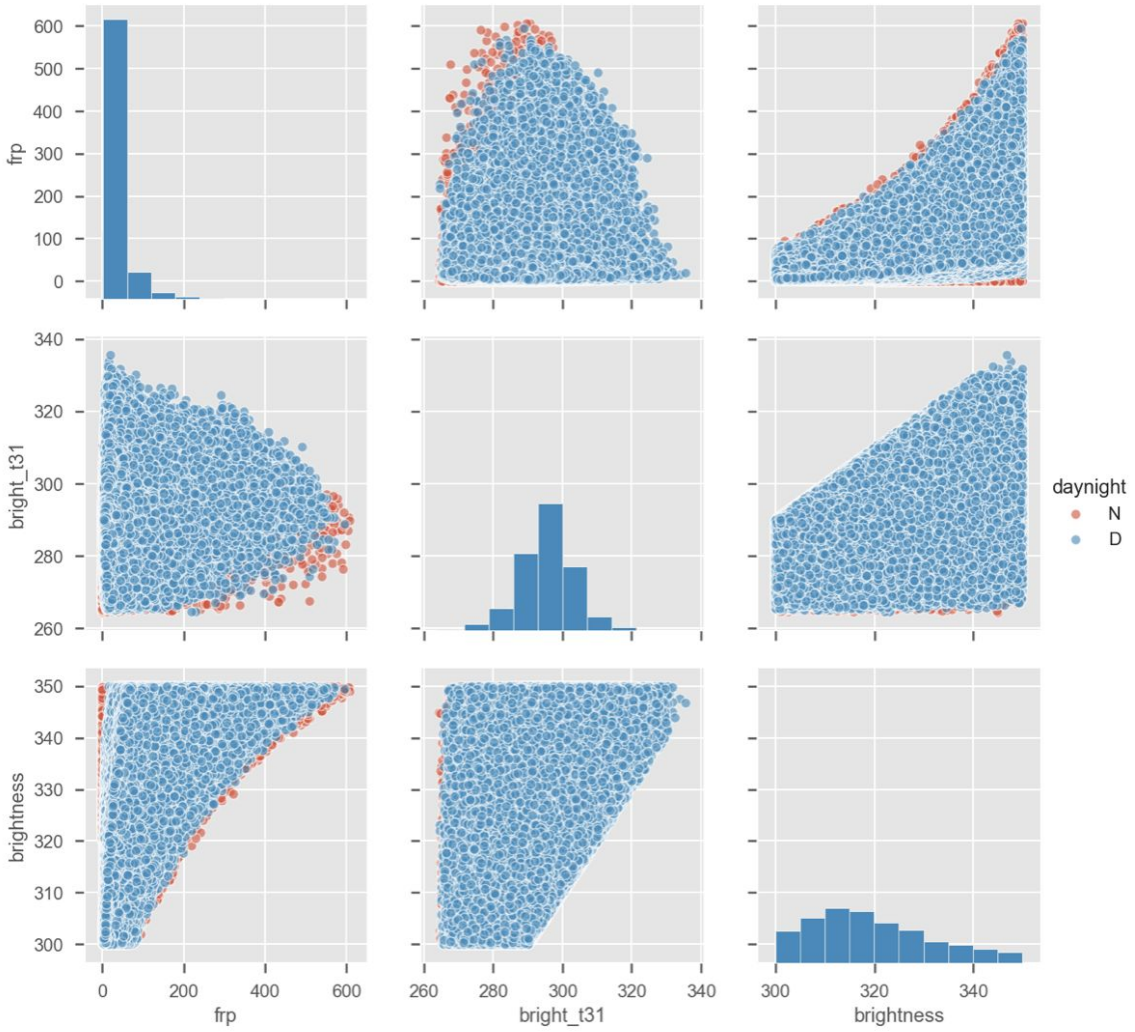
Statistical Analysis Part I

Our initial statistical analysis of NASA key intensity attributes revealed no strong correlation between these attributes. **See pairplot in Figure 2**. To designate the final attribute that would represent fire intensity feature in our model, we further referred to distributions of these attributes (**see Figure 3 and Figure 4**) as well as referencing research. While researches clearly supported that there was a strong relationship between FRP and fire size, FRP and biomass consumption, or FRP and CO2 emission[1], NASA Earth

[1] https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2009JD013769 | https://feer.gsfc.nasa.gov/projects/emissions/ | https://www.fs.fed.us/rm/pubs_other/rmrs_2008_ichoku_c001.pdf | https://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-77862017000200255 |

documentation of MODIS and VIIRS data collection suggested that Brightness would be a more reliable representative of fire intensity for the years 2003 through 2015.[2] Additionally, as seen in histograms, Brightness suffered less data skewness compared to FRP; which made it a more robust variable to use in our model.

Figure 2 - Pairplot of Intensity Attributes



R2 Correlation of Brightness and FRP: 0.415403183
R2 Correlation of Brightness and Bright_t31: 0.335916857
R2 Correlation of FRP and Bright_t31: 0.100193615

[2] https://earthdata.nasa.gov/faq/firms-faq

Figure 3 - Distribution of Fire Radiative Power

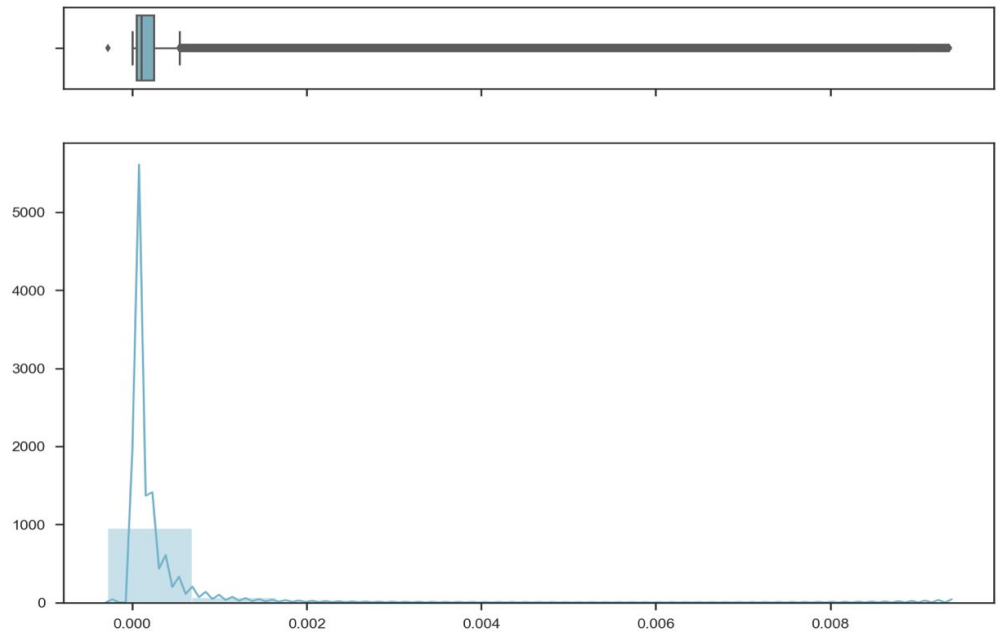
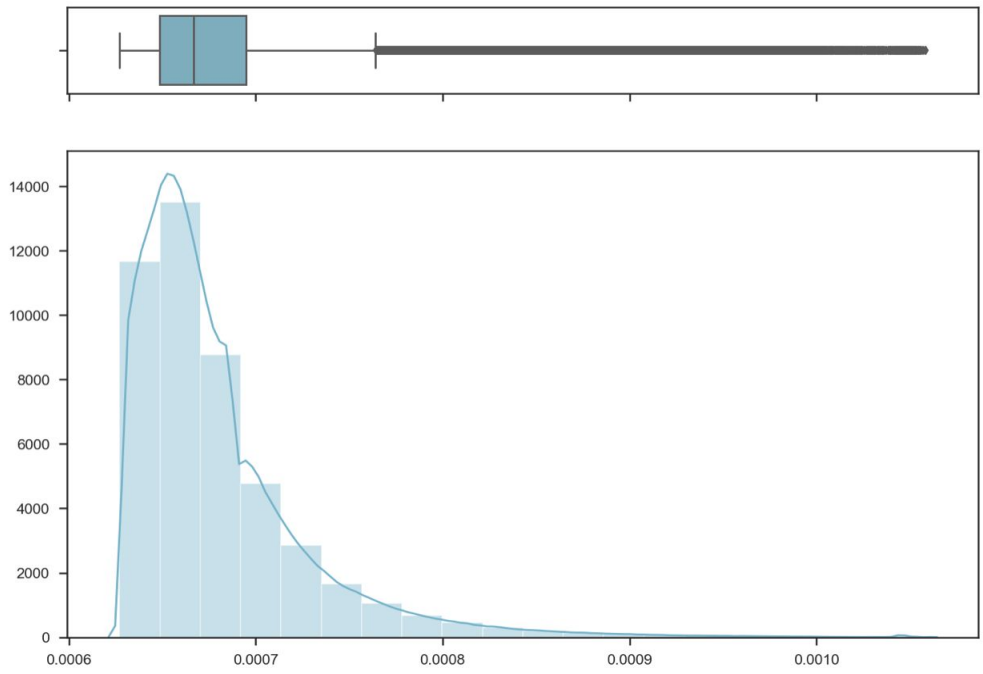


Figure 4 - Distribution of Fire Brightness

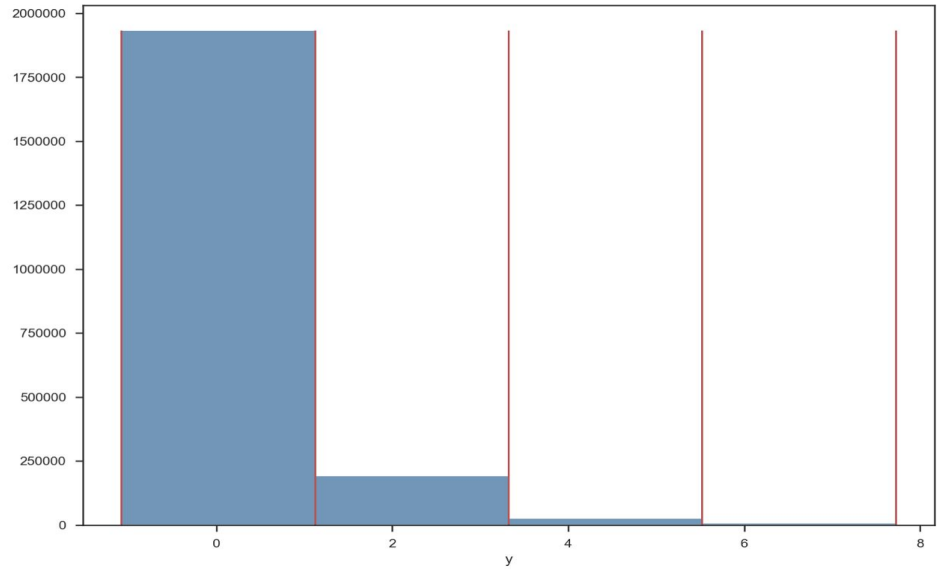


Binning Brightness as our Y variable

The team decided to use classification models to predict the intensity of fire for which the Y variable needed to be binnerized. We used BalancedBinningReference() from Python sklearn library to bin the Y data. The function suggested 4 bins as shown in **Figure 5**.

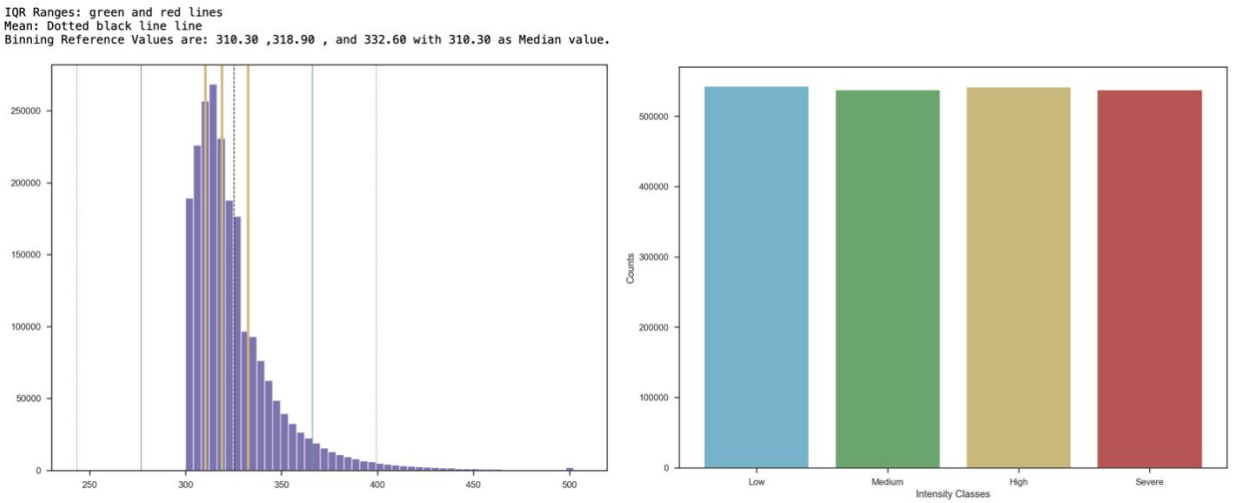
Figure 5 - Initial binning of the Y variable

```
# Balanced Binning of Transformed Brightness
visualizer = BalancedBinningReference()

visualizer.fit(TransformedBrightness)        # Fit the data to the visualizer
visualizer.show()
```



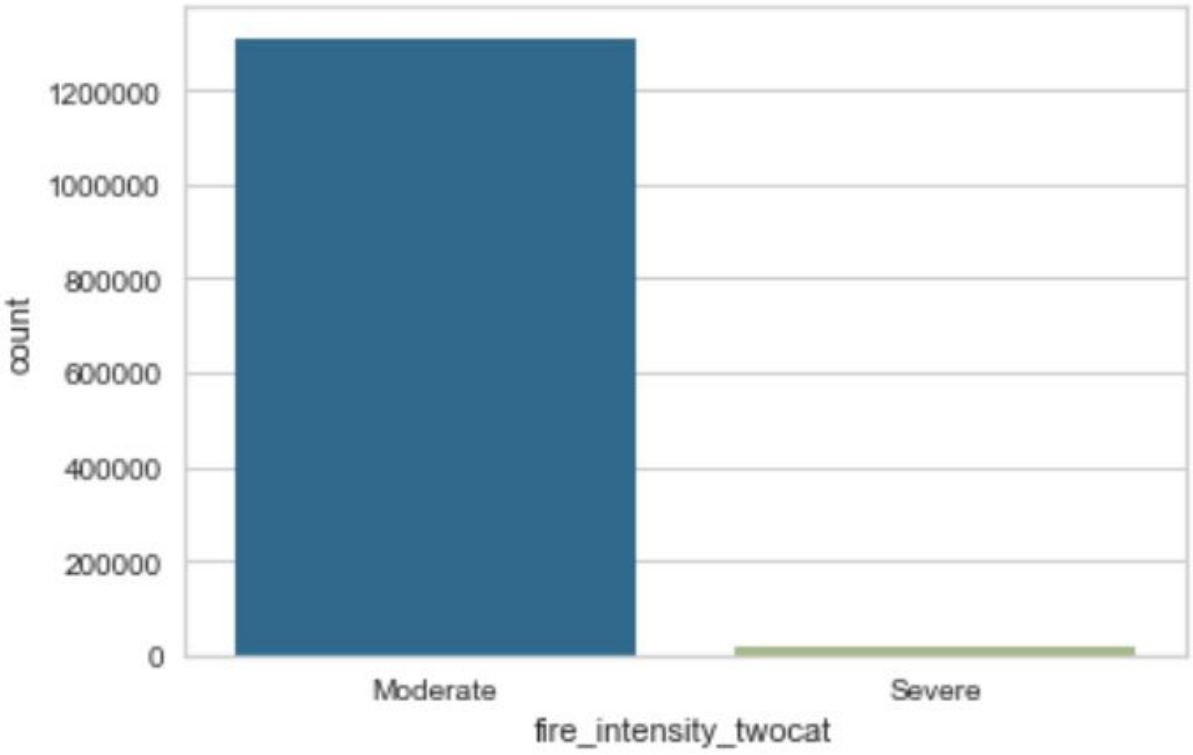
In order to balance the data in each bin, we used Interquartile Reference points as binning references (**Figure 6**)

Figure 6 - Brightness IQR as binning reference values



During the process of model selection and score evaluation, and in order to increase the running performance of the model, the team decided to run the final model with two bins, labeled as “Moderate” and “Severe” (**Figure 7**).

Figure 7 - The 2-bin distribution of the Y variable



To begin our Exploratory Data Analysis, we matched the Longitude and Latitude of NASA fire points against a USDA dataset containing unique records of fires across the continent of the United States to track geographical and seasonal trends. Merging of NASA satellite-observed Near Real Time fire intensity data and USDA land-observed fires enabled us to capture the spread of the Brightness feature in various states. **See Figure 8**. Following steps have been taken to merge these datasets:

1. We used the initial cluster centroids resulted from the DBSCAN of the full NASA spatial dataset.
2. Encoded Day of Year as a Cyclical Feature in both datasets.
3. Utilized the KDTree algorithm to find the nearest unique fire in the USDA dataset to any given point in the NASA dataset.

Below are the respective Jupyter Notebooks:

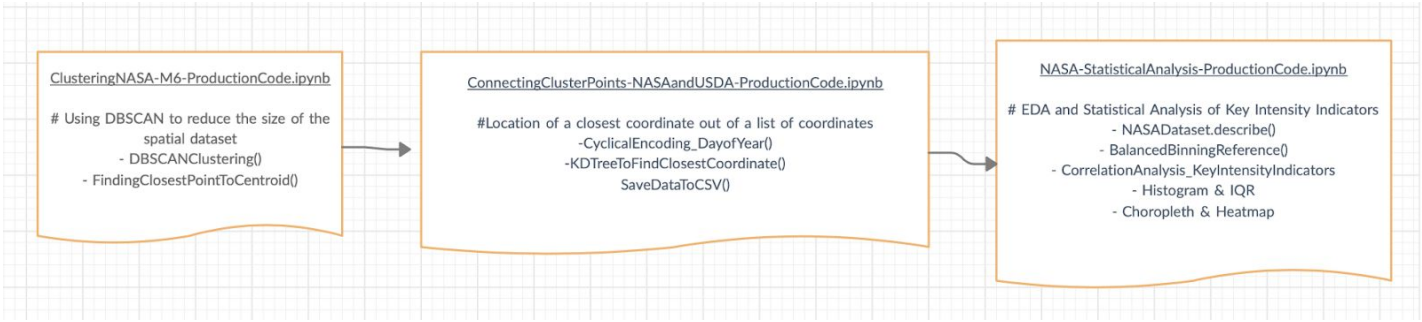
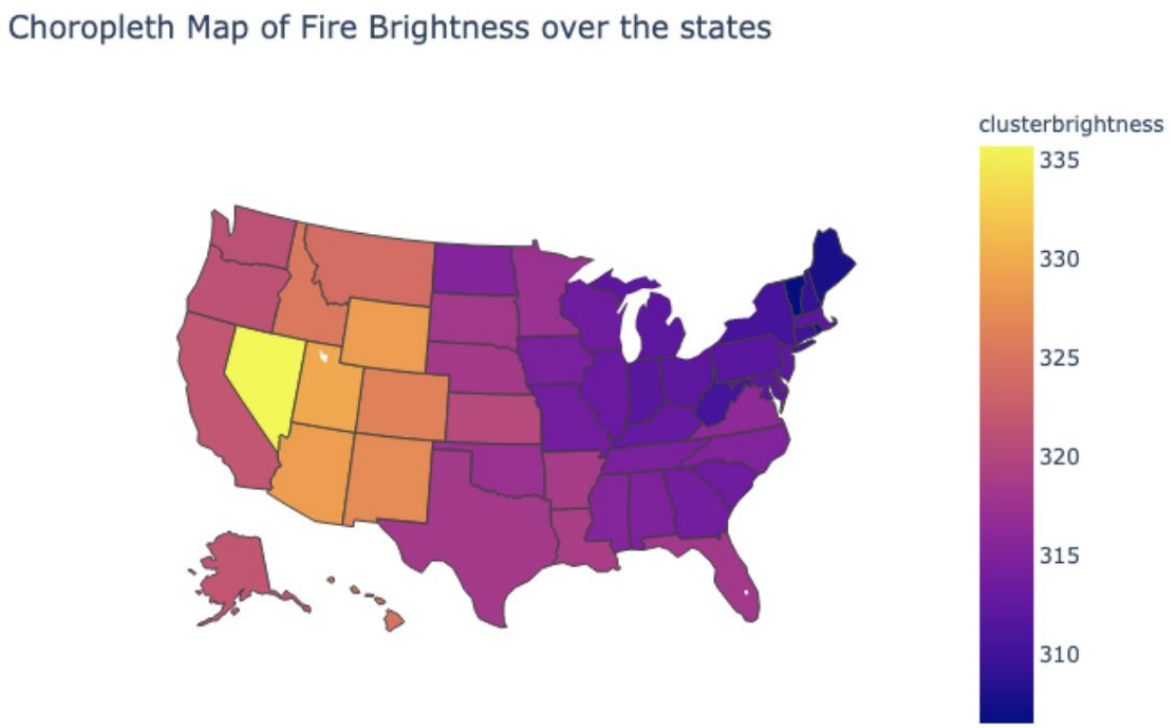


Figure 8 - Choropleth Map of Fire Brightness



Further analysis of this merger suggested that fire danger was more prevalent in certain Latitude and Longitude as well as certain seasons imposing certain weather conditions. **See Figure 9.** Both these findings were later supported by our feature analysis.

Figure 9 - Heatmap of Fire Brightness

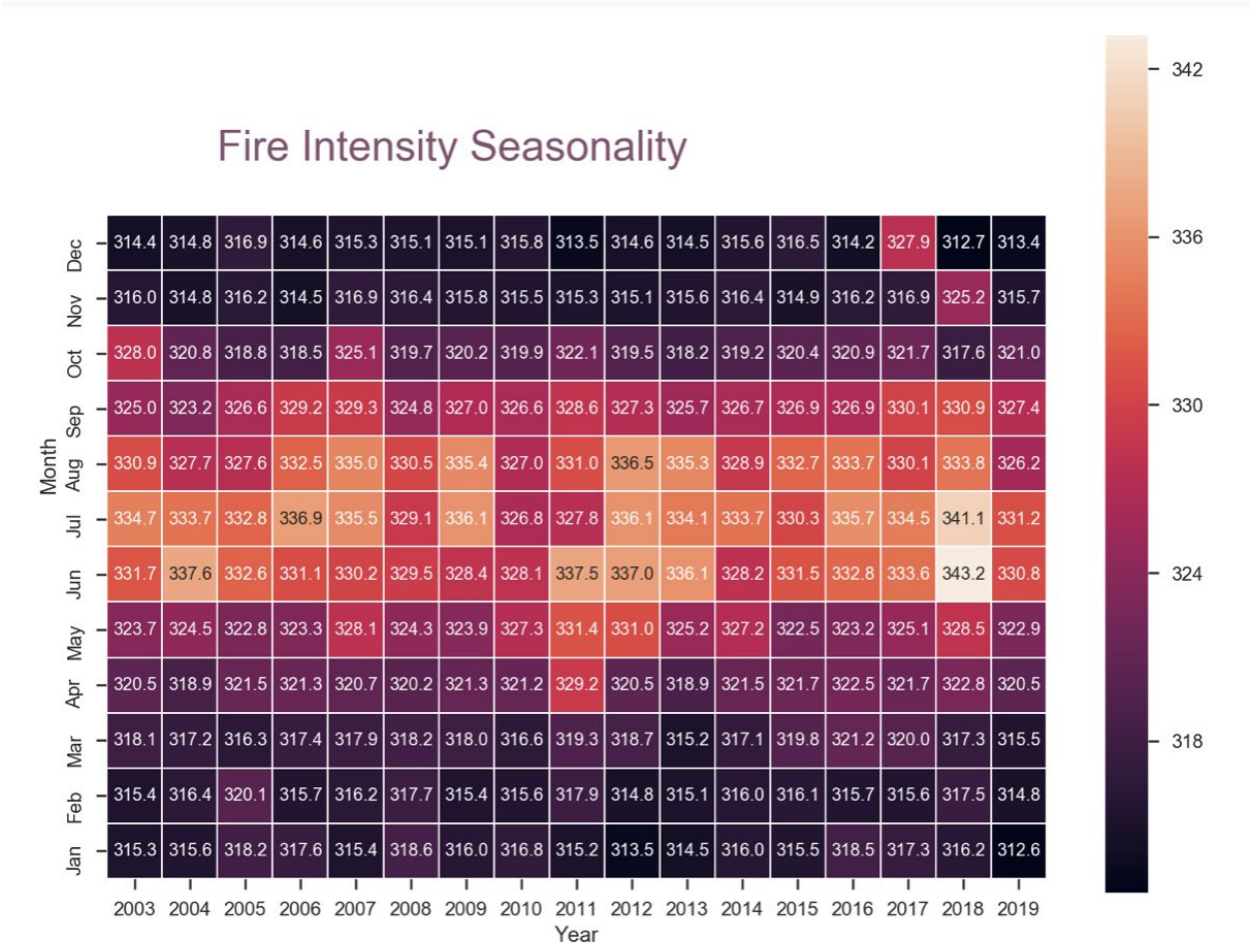


Figure 7 - Cat Plot of the Effects of Seasonal Changes of Temperature on Fire Intensity

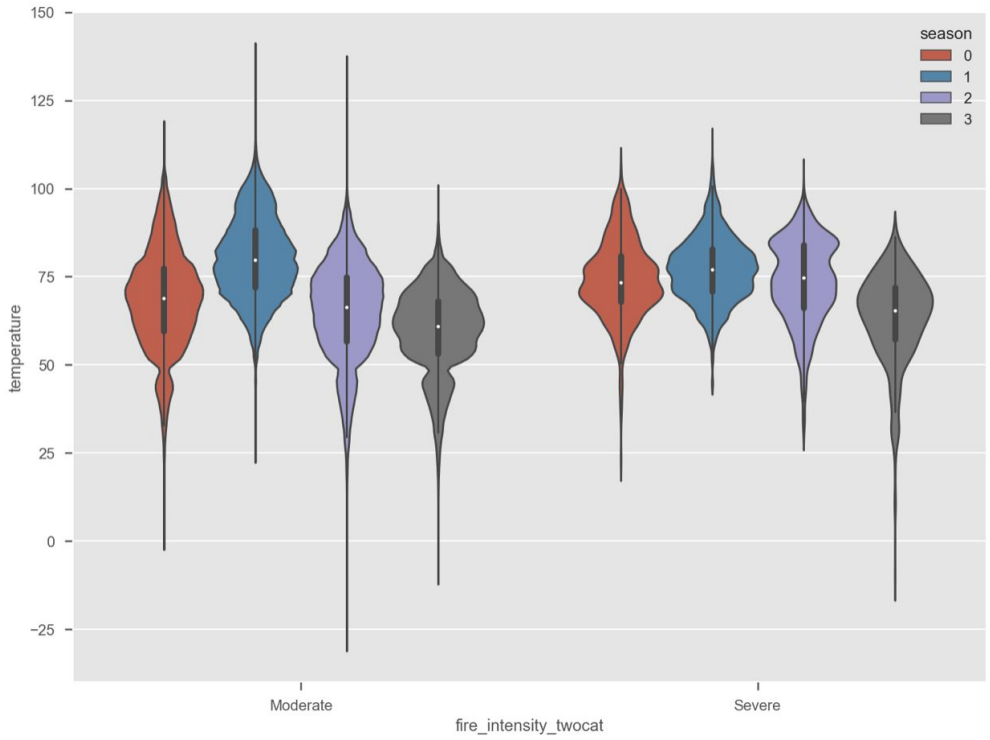
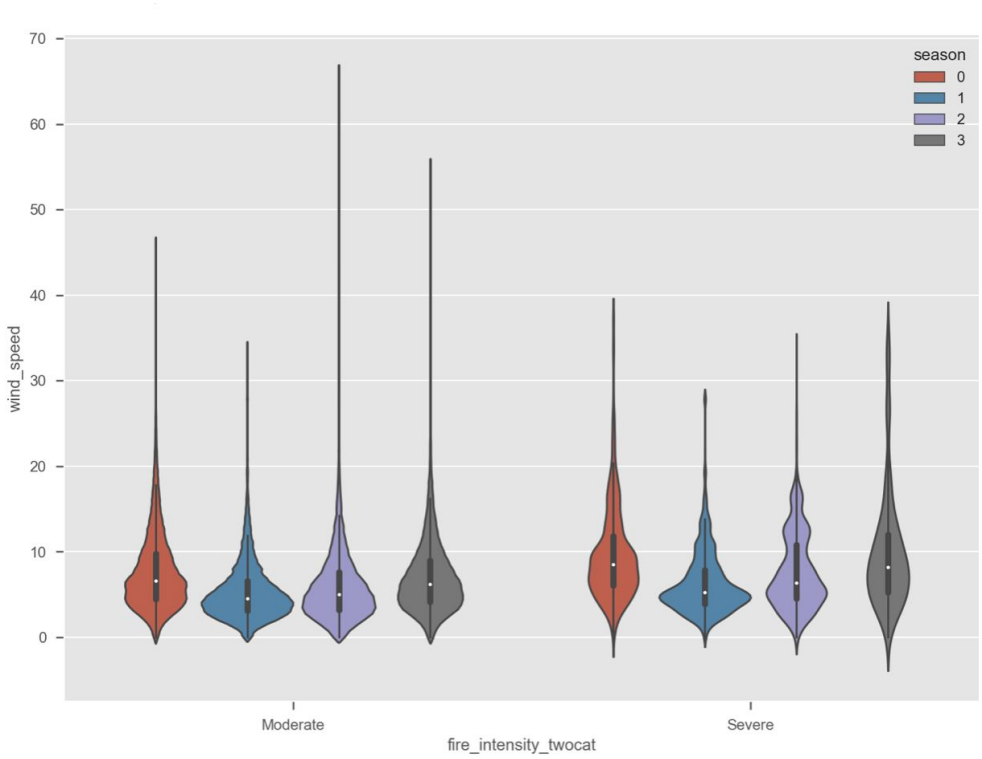
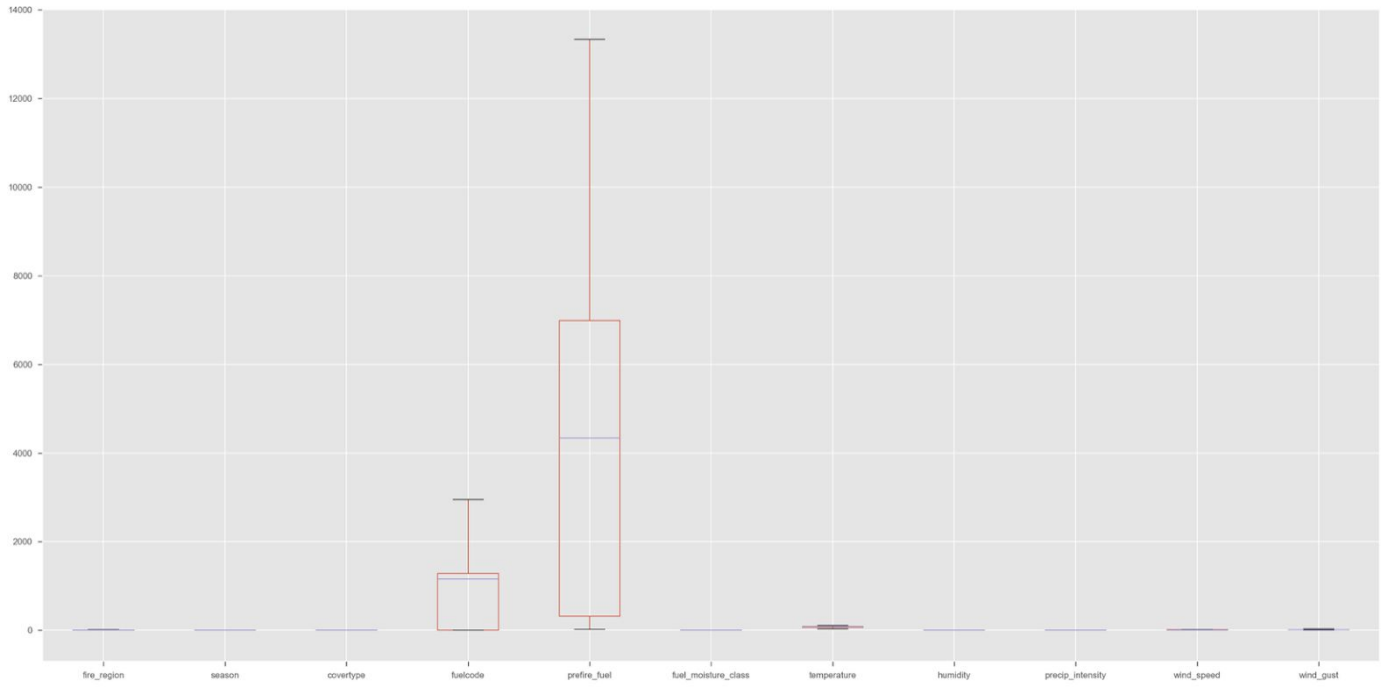


Figure 8 - Cat Plot of the Effects of Seasonal Changes of Wind Speed on Fire Intensity

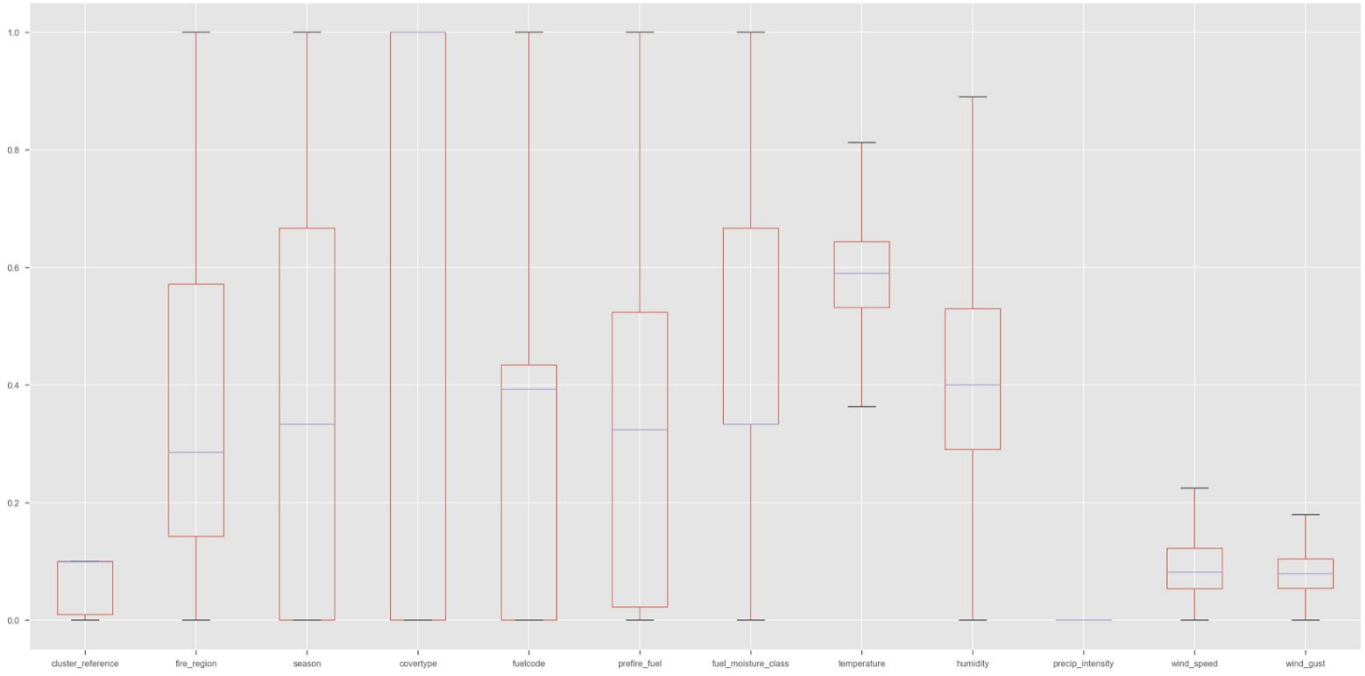


Univariate Analysis of all the X Features

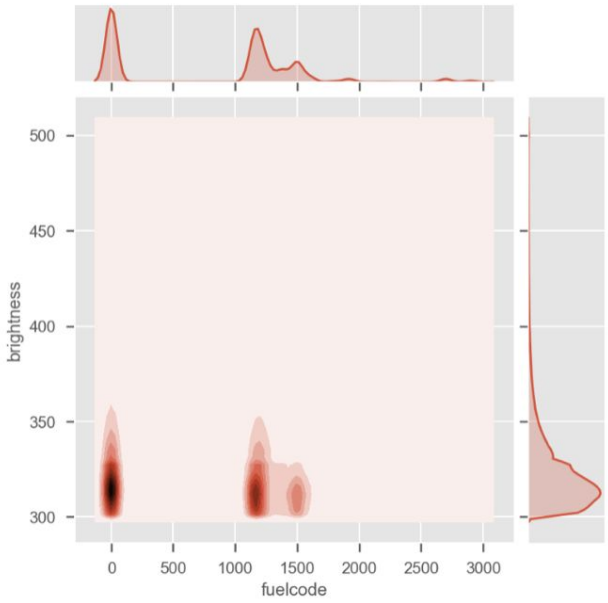
Understanding the shape and spread of the data:

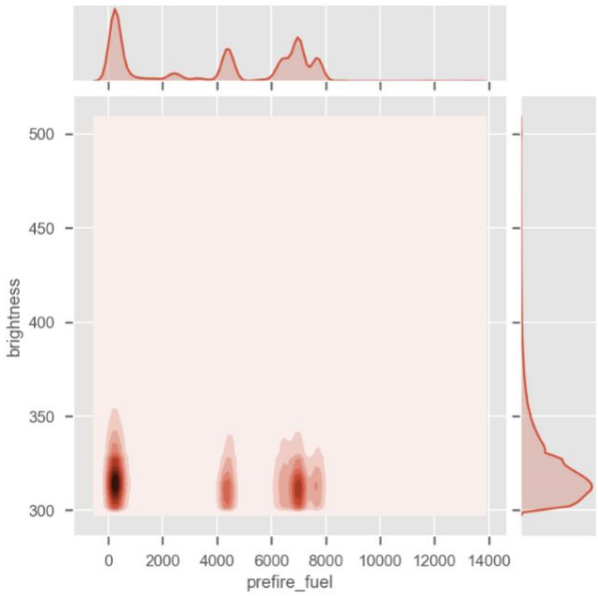


Scaling the data using MinMaxScaler:



Bivariate Analysis of Fuel and Intensity





Statistical Analysis Part II

The second phase of the EDA undertaking built upon earlier data wrangling and munging efforts to identify and label features that were important to future machine learning modeling efforts. EDA part II analyzed unique fire intensity data identified earlier (see Munging and Wrangling Part VI: Fire Brightness Model Input) focusing on a data frame containing key NASA fire intensity features, cluster reference labels, and quartile breakdowns for Fire Intensity denoted by Low, Medium, High, and Severe categories.

As a matter of information, the Jupyter Notebook containing the analysis in EDA Part II is in the Predicting Wildfires repository in the notebooks folder called 'Fire Intensity Statistical Analysis Production Code.'

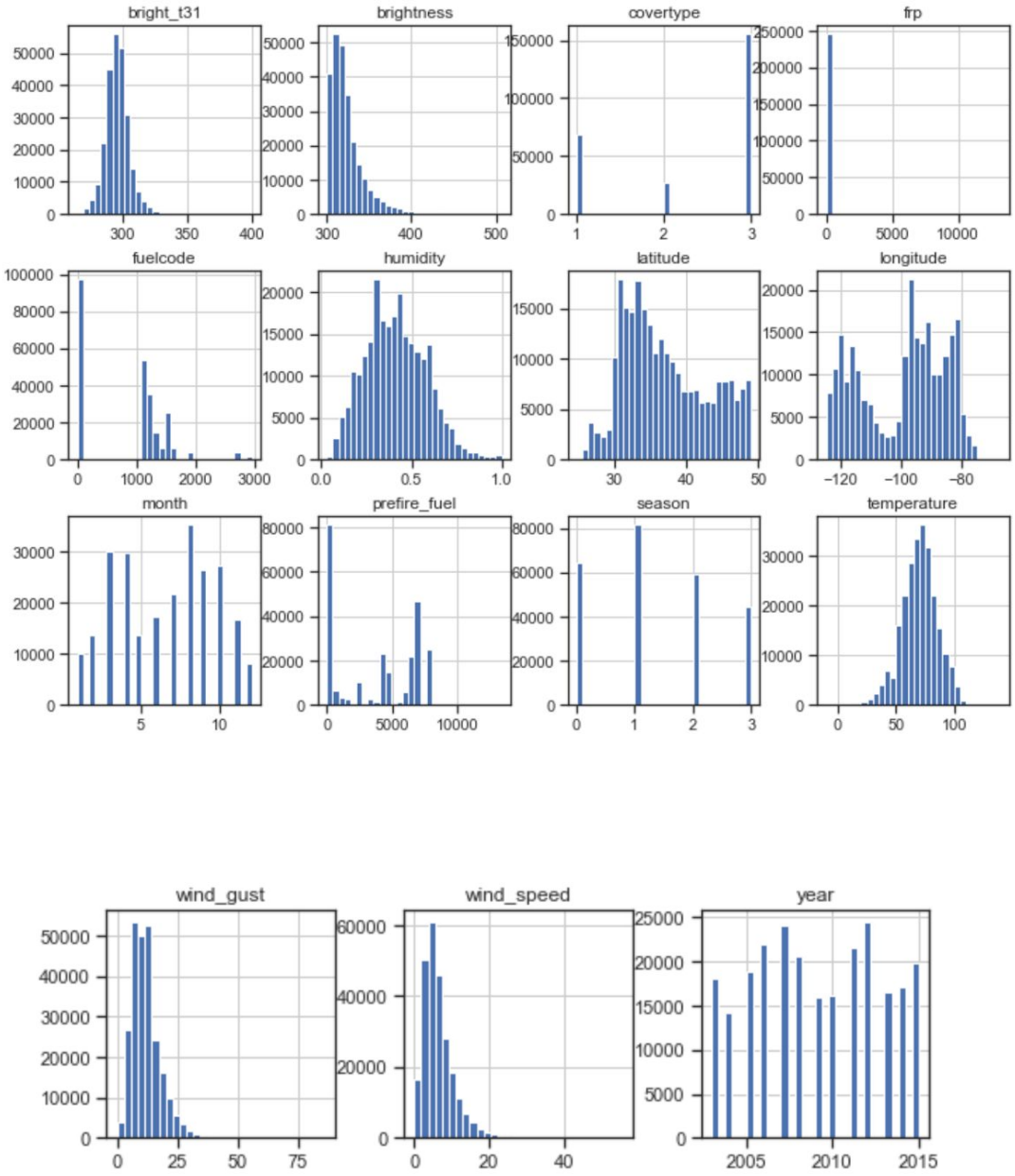
Fundamental to our visual exploration efforts was generating histograms to display the shape and spread of the model features. Based on our analysis, we observed that the data was scalable and most of the features had minimum and maximum values that one would expect to see with the features affecting fire intensity.

In [13]:
```
# These features all increase with each other - no inverse relationships
emstat[['wind_speed', 'wind_gust', 'brightness', 'bright_t31']].agg(['mean','min','max'])
```

Out[13]:

	wind_speed	wind_gust	brightness	bright_t31
mean	6.362801	11.673115	323.864929	295.840523
min	0.000000	0.000000	300.000000	264.500000
max	55.250000	85.990000	505.800000	400.100000

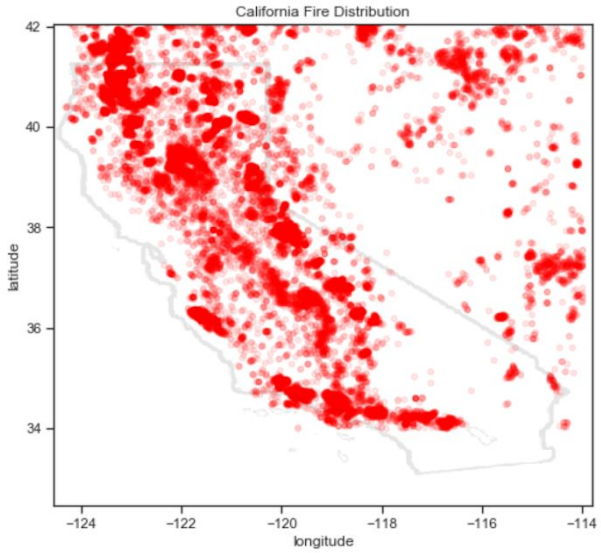
A view of the histograms resulting from the Fire Brightness Model dataset are shown as follows:



In the case of brightness (the 2nd tile from the top-left) the data falls within the range of 300 to 505 with a mean of 323. The frequency distribution is not symmetrical and is skewed to the right indicating the data is more oriented toward the mean. As a point of reference, brightness is the temperature of the fire pixels measured in Kelvin as collected from NASA's Moderate Resolution Imaging Spectroradiometer satellite.

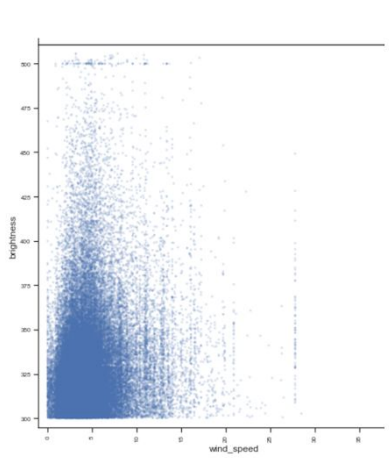
Data clustering was central to visualizing wildfire intensity and using the Density Based Spatial Clustering of Applications (DBScan) was an effective approach to defining and labeling unique fire events. Our team determined unique fire clusters for the years between 2003 and 2015, capturing the centerpoints for each cluster, and aggregated the fire clusters for all of the states in the United States. For the following depiction of the state of California, we used a plotting functionality oriented on a thin wrapper using the Matplotlib library.

```
In [20]: # California Fire Distribution Map

california_img=mpimg.imread('CaliforniaMapOutline.gif')
emstat_west.plot(kind="scatter", x="longitude", y="latitude", c= "red", figsize=(10,7), alpha=0.1)
plt.imshow(california_img, extent=[-124.55, -113.80, 32.45, 42.05], alpha=0.1)
plt.title('California Fire Distribution')
plt.show()
```

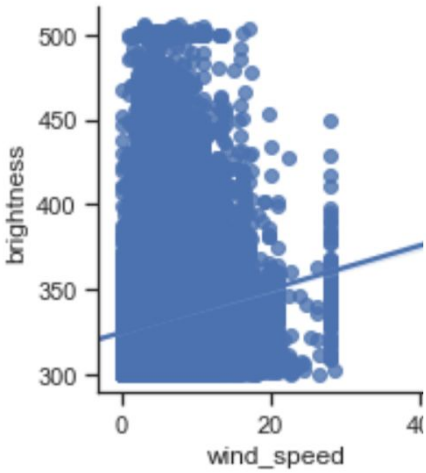


Throughout the EDA process, we extensively used Seaborne and Matplotlib statistical tools to illustrate how the brightness feature is loosely concentrated around the median and is a function of wind speed. Looking at the following graphs, one can see that the visualization improves from good with the scatter matrix, to better with the pair plot, and best with the joint plot. The plot on the right reflects the best granularity between the wind speed and brightness correlation. And finally the R2 value shows a weak regression and you can see this in the three plots.

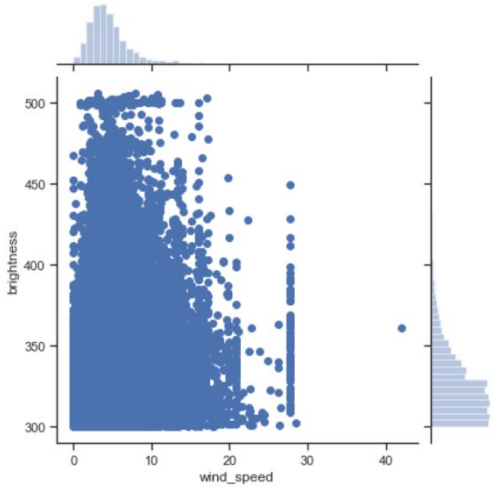
R2 Correlation of windspeed and brightness: 0.018661473



Scatter Matrix



Pair Plot



Joint Plot

In [26]:
```
# Correlation Analysis of key intensity indicators part 2
x_values = emstat_west['wind_speed']
y_values = emstat_west['brightness']
z_values = emstat_west['fuelcode']

correlation_matrix1 = np.corrcoef(x_values, y_values)
correlation_matrix2 = np.corrcoef(x_values, z_values)
correlation_matrix3 = np.corrcoef(y_values, z_values)
correlation_xy = correlation_matrix1[0,1]
correlation_xz = correlation_matrix2[0,1]
correlation_yz = correlation_matrix3[0,1]
r_squared1 = correlation_xy**2
r_squared2 = correlation_xz**2
r_squared3 = correlation_yz**2

print(f"R2 Correlation of windspeed and brightness: {r_squared1:.9f}")
print(f"R2 Correlation of windspeed and fuelcode: {r_squared2:.9f}")
print(f"R2 Correlation of brightness and fuelcode: {r_squared3:.9f}")
```

```
R2 Correlation of windspeed and brightness: 0.018661473
R2 Correlation of windspeed and fuelcode: 0.118979408
R2 Correlation of brightness and fuelcode: 0.002786746
```

Although there was a weak regression between the wind speed and brightness, we were compelled to use as many visualization techniques as possible to portray the fire intensity feature variance, as well as enhance our learning of the myriad visualization techniques despite the data's multivariate nature.

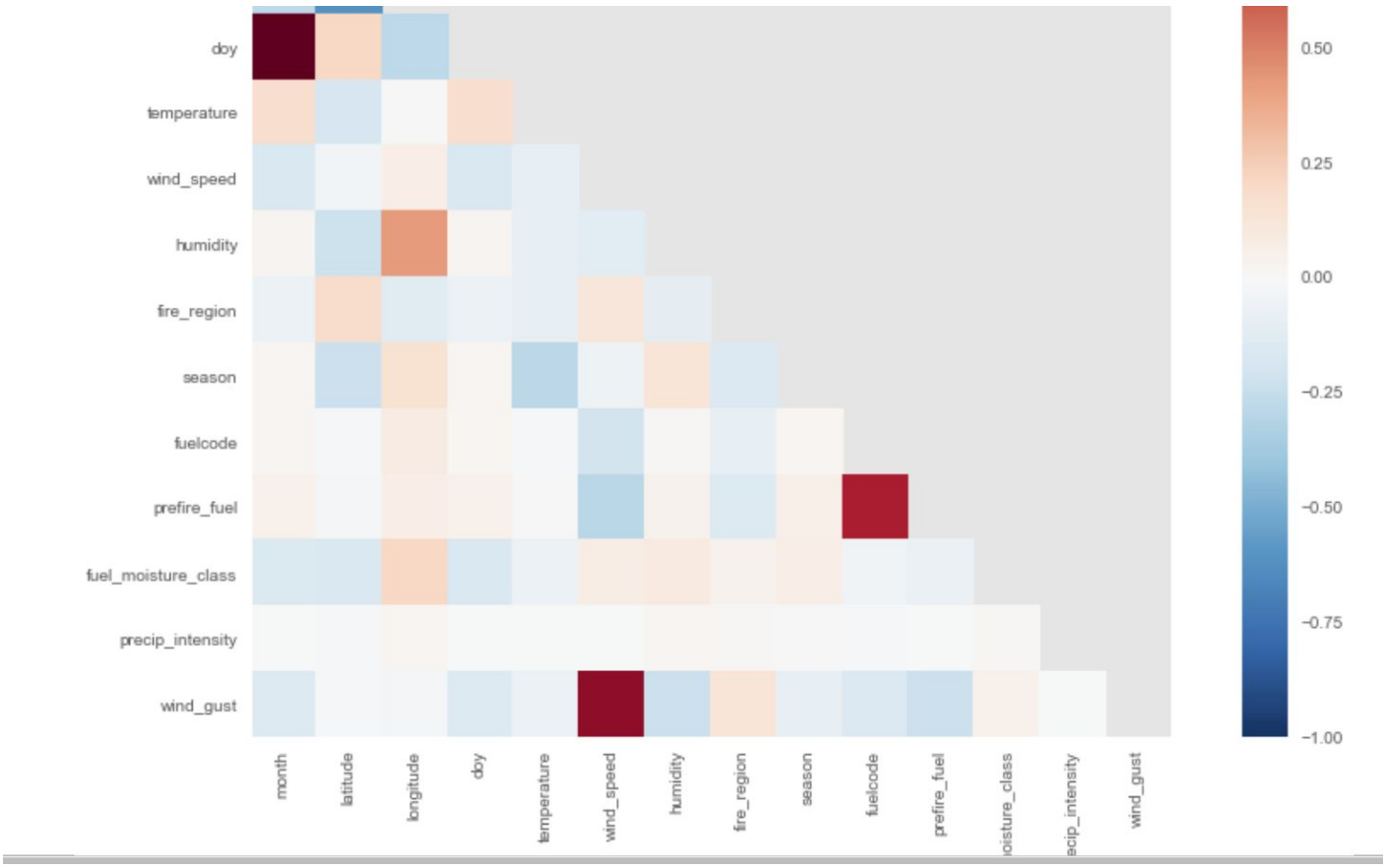
Feature Selection and Modeling

Based on review of the data available to the team through our three primary data sources, we segregated the X features into three primary categories:

1. **Landscape** - Descriptive categories that identified the type of forest/land for the lat/long coordinate record, mass of bio pre-fire fuel present at the location, and a moisture category representing the moisture of the underlying biomass at the location. We felt strongly that each of these features would have a strong correlation with the Y target value; however due to the similarities in Cover
2. **Locational/Time** - With having geospatial as well as date/time information connected to the NASA satellite imagery records (where we sourced the fire intensity readings for each record), we believed that both the geographical location as well as day of year or related season of the fire event would have a correlated effect with the observed fire intensity. With that said, we selected [latitude, longitude, and the derived KMeans Regions] as features for the Fire Brightness model. Our initial view was that lat/long may be overly complex for the model, therefore we moved to derive a regional view of the country using KMeans to determine 8 region categories for the entire dataset. Upon further feature analysis and given the fact that our final shortlist of models were all Tree Based (that would not require normalization of numerical features), we found that individual lat/long provided a higher Pearson Score than using the Regions features. Secondly, we viewed date and time two ways, the Day of Year (doy) and a determined Season category that we derived from the data, again to provide a more generalized view of the periods. Upon further examination during model testing, we found that DOY provided a stronger data feature score than the regions therefore used DOY primarily for our model going forward.
3. **Weather** - was not readily available in the datasets. The DarkSky API was used to match local weather conditions to the existing fires. The primary fields used in weather that proved to be predictive and readily available were temperature, humidity, wind, wind gust, and precipitation. The hypothesis was weather conditions would greatly impact wildfires. Ideally weather several days before a Wildfire would have been pulled but this was not possible due to time constraints. The weather the day of the event will be used.

The Pearson Ranking was used to help identify high multicollinearity among features. This was an important tool that allowed us to remove features that complicated the model and did not contribute to the predictivity of the model. After analysis two sets of highly correlated features were kept. Fuelcode is categorical and relied on numerical codes to represent the categories and Prefire fuel is a continuous value of biomass on the land. Fuelcode would be one hot encoded so likely not to remain highly correlated with the Prefire fuel mass amount. Wind and wind gust were both kept after analysis they did not highly impact the model having both. Overall the features were not highly correlated after deconfliction. See figure 9 below.

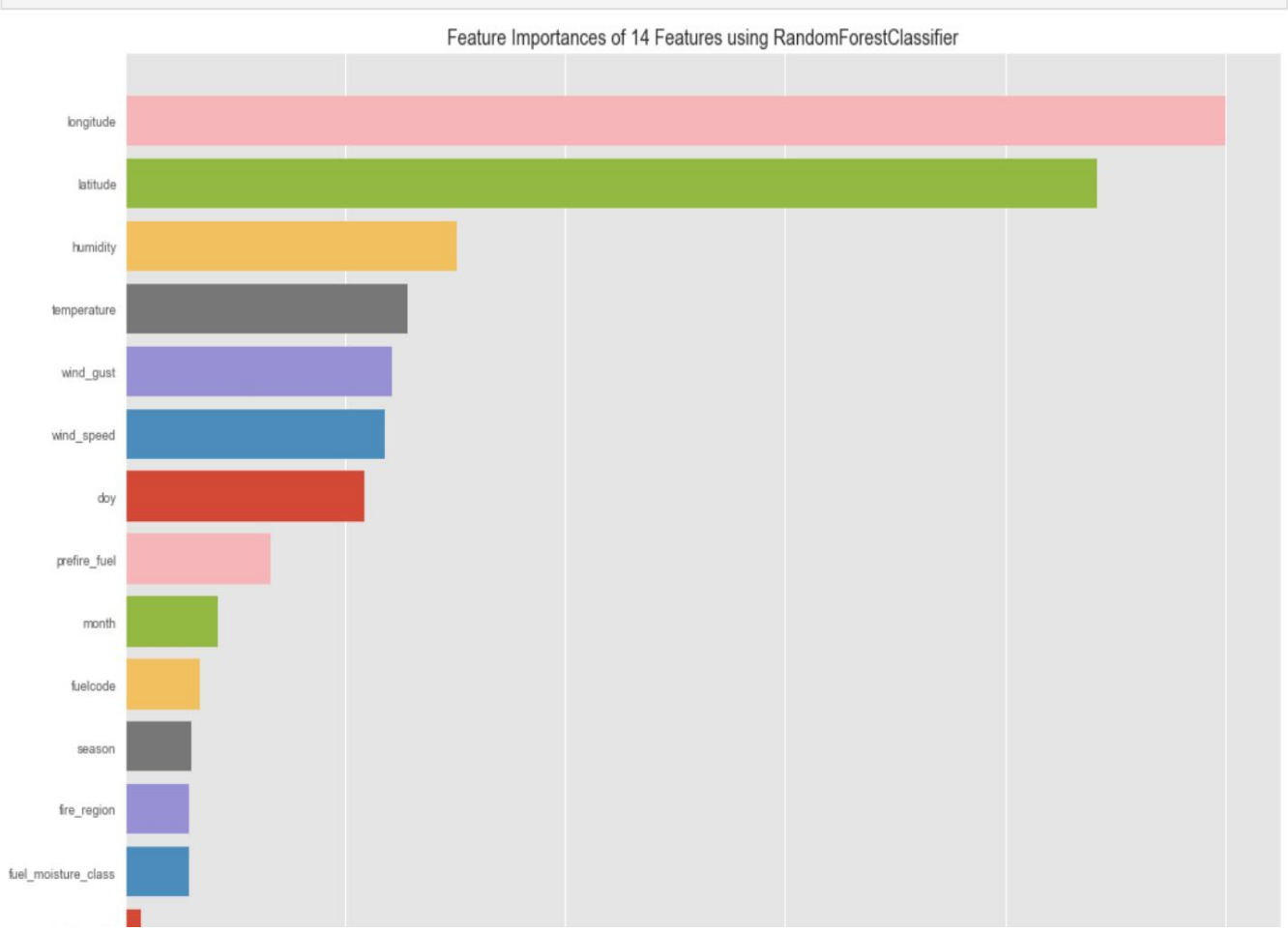
Figure 9 - Pearson 2D Ranking



Feature Importance

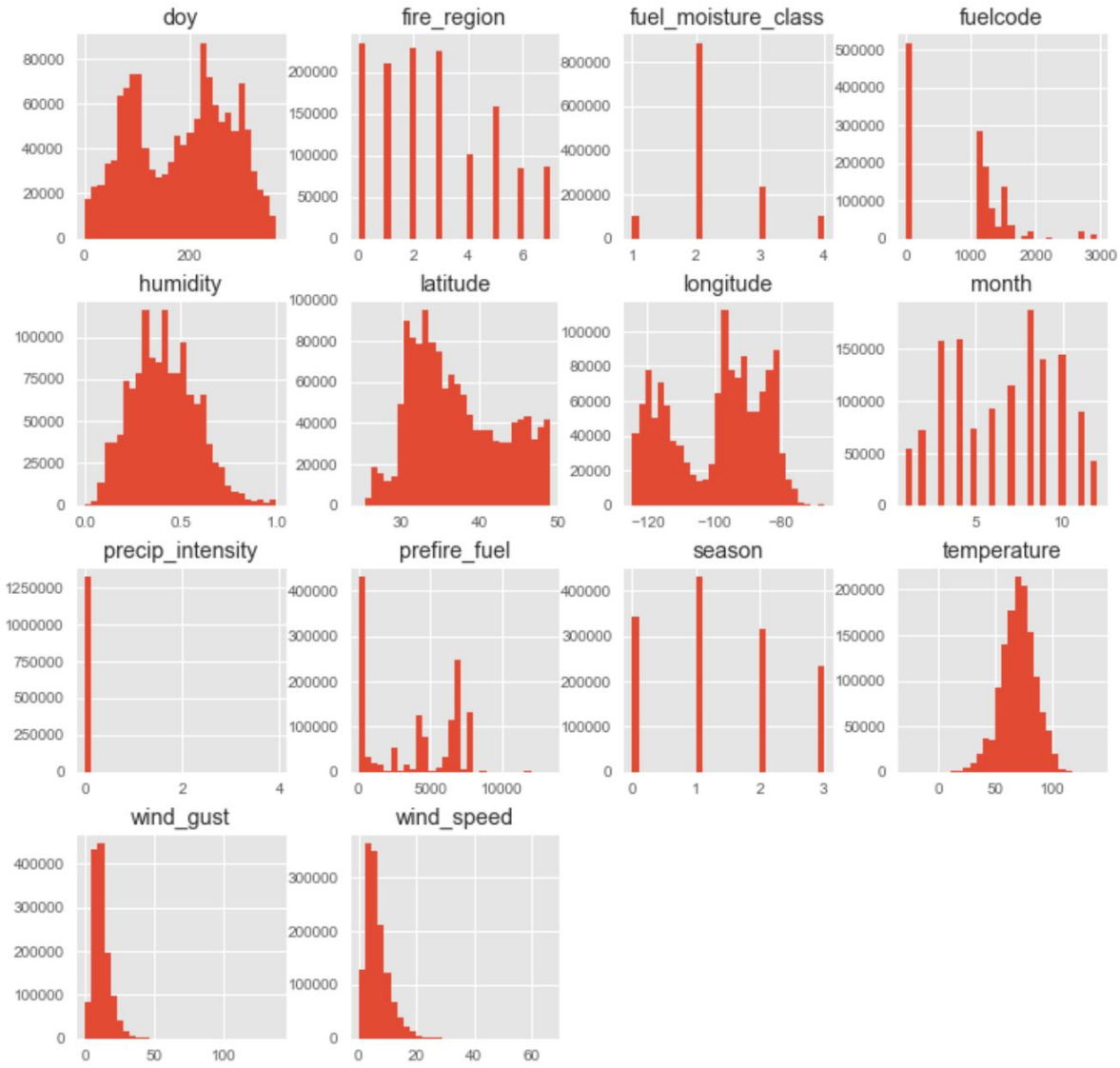
The main features centered around location, cover type, time/season and weather. The feature importance helped to show which features would be most predictive. Season and fire_region features were created to help minimize the complexity of the model but they did not score as well as other features. Duplicative fields values such as month were removed as this data could be encoded elsewhere such as in day of year. This was a valuable tool used to simplify the model to about 12 features.

Figure 10 - Feature Importance Ranking



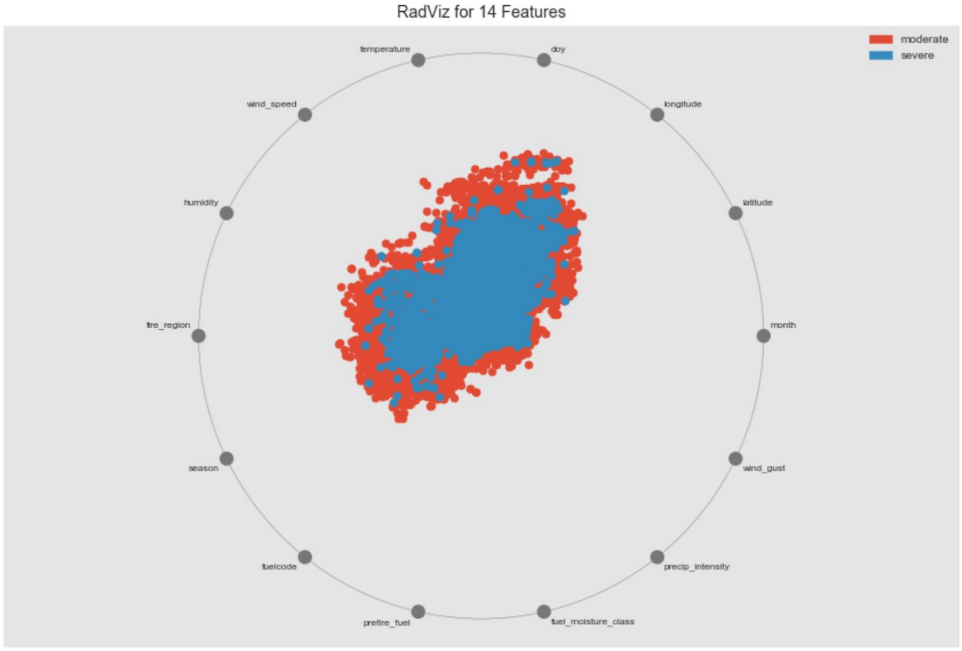
The features were also plotted with histograms. Being a mix of continuous and categorical features no immediate patterns were clearly observable but it helped to verify we once again did not have extremely highly correlated data.

Figure 11 - Feature Histogram



Radviz was also employed to help determine the feature importance. It was slightly skewed towards lat/lon and day of year but not definitively so to be conclusive but it confirmed the Feature Importance ranking that location was a very important predictive aspective of the fires. RadViz can be seen in the visualization below.

Figure 11 - RadViz



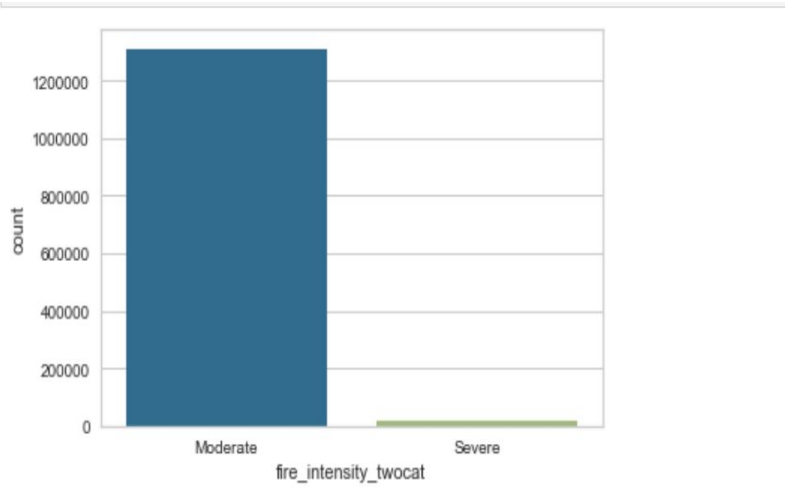
Key takeaways from our analysis:

- One hot encoded needed for certain features like fuel_moisture_class, fuelcode
- Numerical Scaling needed for weather, lat/lon, temperature, humidity,etc.. as these were on a very large and differing scale
- The models did not rank our generated categories highly such as season and fire_region. Ironically these were generated to help the model but I believe our fields may have been too broad. So instead of 4 seasons maybe 12 months would have been more predictive as the middle of summer is usually very different from the end of the summer. Or it could mean summer is very different in different parts of the country so that may not be a good way of predicting what time of year means.
- Settled on 12 features and went back on forth on wind gust and precipitation as being valuable based on research and intuition. These features were ranked medium to low by the algorithms.

Machine Learning

ExtraTreesClassifier was selected after analysis. The major pros of this classifier was it had a high recall for Severe fires. The project contains a large number of instances at 1.3 million but it is heavily skewed to just one class. There was a great class imbalance that led to the models performing poorly.

Figure 12 - Bin imbalance



SMOTE was used to generate synthetic data and create equal bins. However too much synthetic data created a bias and the model was only able to predict fires on synthetic data and not real data. Undersampling majority classes and only slightly oversampling minority classes proved to be most effective.

In the end we settled on a 5:1 ratio of real moderate fires (100K) to (20K) Severe fires in our dataset. From the 20K, 7K was generated to bulk up the test dataset An additional 4K of Severe fire data was left over for testing once the training of the model was completed.

We explored a few linear models and several ensemble classifiers were evaluated to include RandomForest, AdaBoosting and Bagging. They performed fairly well but did not equal the recall of the ExtraTreeClassifiers which was also a bit faster.

SVC did not perform very well. KNeighborsClassifier had a good F1 score but was very slow and expensive to run.

Figure 13 - Grading

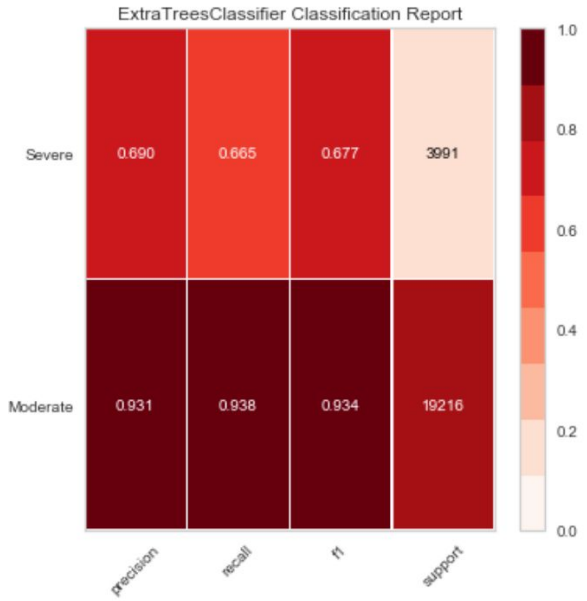
Models Evaluated	Findings	Further Evaluated
MultinomialNB()	Lower F1	No
LinearSVC()	Lower F1 -	No
KNeighborsClassifier()	Higher F1 / Slow	Yes
RandomForestClassifier()	Higher F1 -	Yes
ExtraTreesClassifier()	Higher F1 -	Yes
BaggingClassifier()	Slow, Complex	No
AdaBoost()	Lower F1, Poor Recall	No

Key findings:

- Extratrees and Randomforest performed equally. Extratrees was selected as it performed faster.
- KNeighbors was interesting because it was a simpler model that performed well. It was just slow and expensive to run. Overall it nearly equalled Extratress.

The ExtraTrees F1 scores are below. The major weakness of all of the models was in predicting the Severe fires due to limited amount of data or the makeup of data. ExtraTrees was selected because it had a slightly higher recall. Since we wanted to do the best job possible in predicting a higher impact event over a lower event, we focused on recall even if our precision score was not as high.

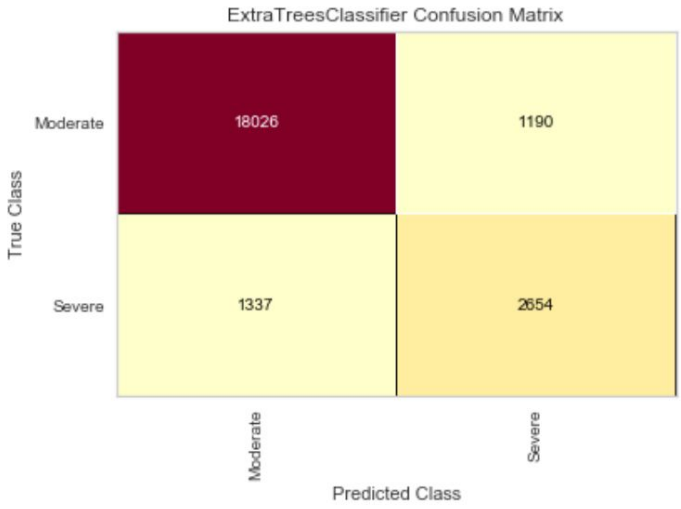
Figure 14 - Grading

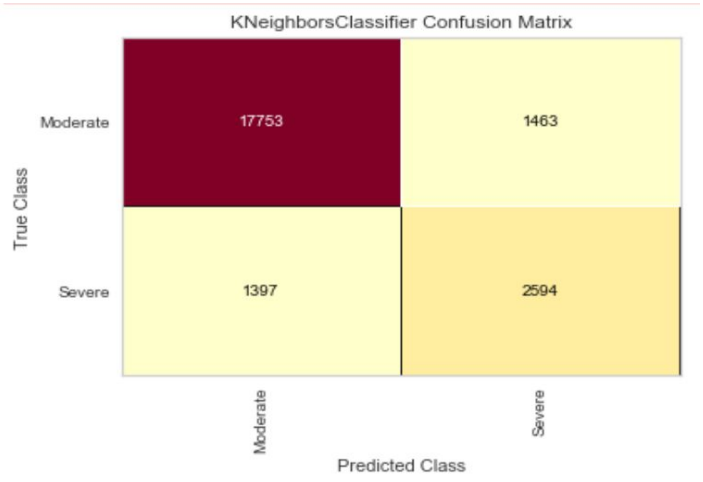



```
Pipeline : Original data F1 Micro Score : 0.8911104408152711
Pipeline : Original data F2 Macro Score : 0.8059854605438945
              precision    recall  f1-score   support

    Moderate       0.93      0.94      0.93     19216
      Severe       0.69      0.66      0.68      3991

    accuracy                           0.89     23207
   macro avg       0.81      0.80      0.81     23207
weighted avg       0.89      0.89      0.89     23207
```

The Confusion Matrix also confirmed the finding as it shaded in yellow the recall and precision for Severe fires could be improved. But in this case we would rather have a high recall and lower precision. Figure 15 is the ExtraTrees Confusion Matrix and Figure 16 in the KNeighbors. They are very close but the computation time of KN was very slow.

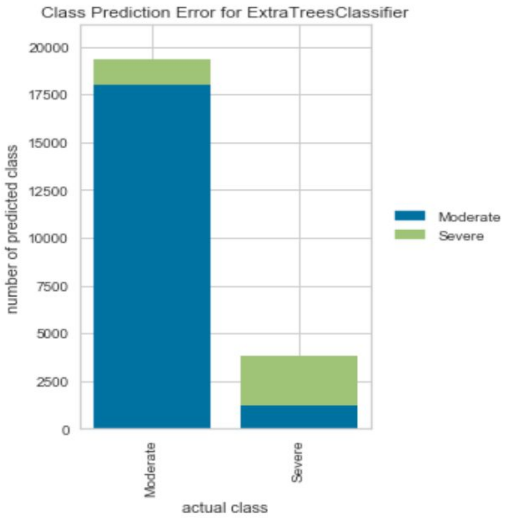
Figure 15 - Confusion Matrix





The class error predictor was another good visualization to show the challenges in prediction Severe fires while moderator ones were not an issue.

Figure 15 - A Prediction Error

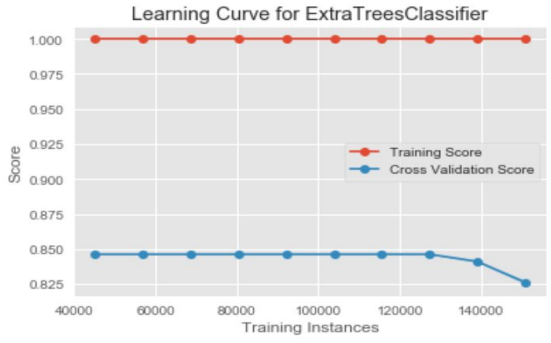


To improve model performance all categorical features were label encoded and all numeric features were scaled.

The model initial estimates for all of the ensemble models show the learning curve was optimal between 100K-200K records. This proved to be fairly accurate and was used to training the model. See figure below.

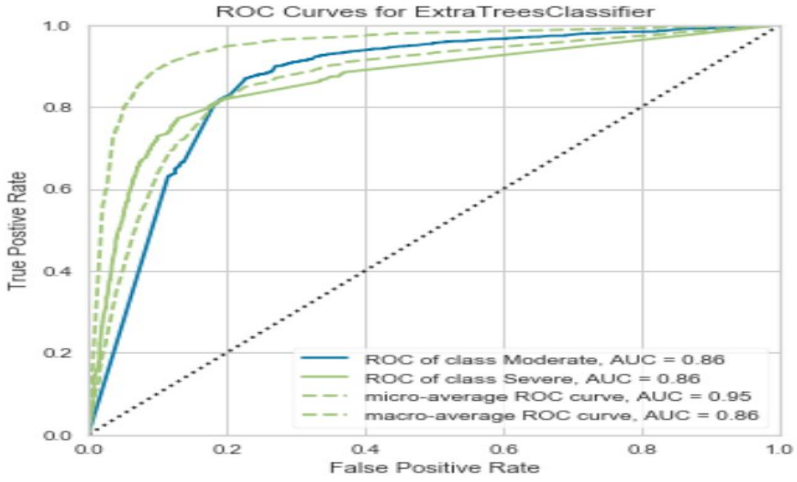
Figure 16 - Learning Curve

```
[30]: # Instantiate the classification model and visualizer
model = ExtraTreesClassifier(n_estimators=5)
visualizer = LearningCurve(
    model, cv=cv, scoring='f1_weighted', train_sizes=sizes, n_jobs=4
)

visualizer.fit(X, y)        # Fit the data to the visualizer
visualizer.show()           # Finalize and render the figure
```



The ROC/AUC graphic is depicted below. The algorithms were able to learn fairly well due to the high number of instances. The severe fire data was more challenging.

Figure 16 - ROC Curve



The concerns for the current implementation is the potential overfitting, standard distribution of data and other unknowns given limited time to learn and investigate the model.

Moderate fires are predicted with precision and recall while Severe only with a clearly better than chance guess but not sure how it compares to human expertise. Since Severe is the most dangerous type of fire so we want to ensure it is predicted even if it generates more false negatives. More data or research would likely be needed to improve the recall and precision of Severe as a future effort.

Hyperparameter Tuning

Hyperparameter tuning for the project was done with GridSearchCV. This yielded mixed results. The majority of the tuning affected the Severe classification Precision and Recall. Tuning the parameters inversely affected Precision and Recall. One change would increase precision but lower recall or vice versa. Below were the recommended parameters for ExtraTreesClassifier.

Figure 13 - Parameter Tuning

ExtraTreesClassifier() Hyper parameter tuning

[6]:
```
# Use GridSearchCV to see whether we need to tune any of the  parameters.
model = ExtraTreesClassifier()

# Create a dictionary with the parameter option'
parameters = {'n_estimators': [5, 10, 25, 50, 100], 'criterion' : ['gini', 'entropy'],
              'max_depth': [5, 10, 25, 50], 'bootstrap':[True,False],
              'oob_score': [True,False], 'n_jobs' : [-1]}

clf = GridSearchCV(model, parameters, cv=5)
clf.fit(X, y)

print(clf.best_estimator_)
```

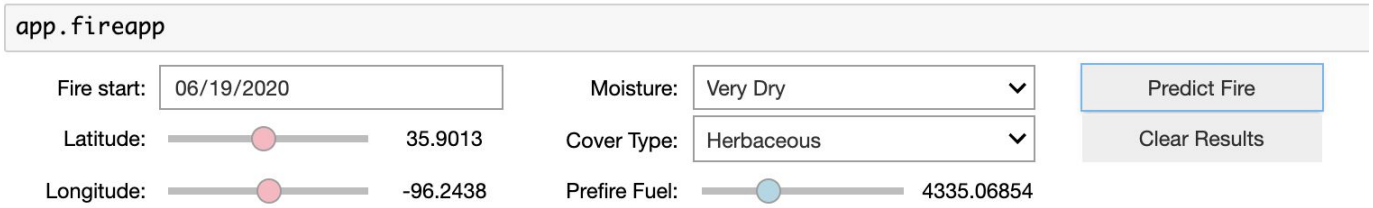
```
ExtraTreesClassifier(bootstrap=True, max_depth=10, n_jobs=-1, oob_score=True)
```

After trial and error the default hyper parameters or setting the base trees or n_estimator to 10 provided the best results for our dataset.

Application

We decided the best way to ship this model to end users would be to create a lite app that would allow for 5-6 user inputs then it would collect any other data inputs from the DarkSky API. This app uses ipywidgets and was initially built in a Jupyter notebook. In order to make the final product more user friendly we migrated the code into a standard python script so you can call the library and execute the app in one step. The inputs from the user ended up being, date, latitude, longitude, cover type, moisture, prefire fuel. The script uses the same DarkSky function to take the date, latitude and longitude that the user enters and pings the api in real time to get the weather data needed for the X inputs. The pickled model is sourced in this script and then returns the prediction of the fire severity.

The app allows a user to save the historical pulls in order on screen but also has a button to clear this data.




```
Warming up...
Finding season...
Pulling weather data from DarkSky...
Model Loaded...
Inputs Loaded...
Calculating Intinsity...

------Model Input Values------
Latitude:  35.9013
Longitude:  -96.2438
Day of Year:  171
Fuelcode:  1
Fuel Moisture Class:  1
Prefire Fuel:  4335.068543
Temperature:  85.72
Humidity:  0.58
Precip Intensity:  0
Wind Gust:  19.64
Wind Speed:  10.67
-------------------------------

----------------
|    Moderate    |
----------------
```

Lessons Learned and Next Steps

As for lessons learned, the team determined quite a few areas including:

1. Time Required for Data Wrangling when the dataset is of a large size.
2. Skewness in your target Y data, as well as other datasets will cause significant hurdles to overcome as you work around the data imbalance.
3. Git-Hub in certain situations was not easy to navigate and there were instances where data and information was lost during the transition of updates across the platform.
4. Project management tools such as Slack and Trello are invaluable tools to support communication, sharing of information and assignment of tasks and tracking project milestones.
5. Refining models and assessing their performance is an interactive process and takes time to move through different variations and ideas.

Next Steps - The team has identified several areas that we'd like to continue to pursue as we further develop our Python, Statistical and Machine Learning capabilities. In the interest of time during the Capstone period, we were unable to overcome certain hurdles that arose during the project including:

1. **DBScan and Initial Cluster Identification** - due to the large datasets under review at the onset of the capstone, we were concerned about allocating too much time to fine tuning this specific clustering process during the early stages of the project. As we continue to assess the project, we'd like to dedicate more time to:
    a. **Adjusting the Min Sample Size,** as well as adjusting other parameters that could more accurately cluster geographic the coordinate data with other criteria such as date/ time.
        i. Upon review of the clusters created, although geographically they were appropriate, we noted that some clusters had very low counts of underlying data points (e.g., 1 coordinate). Per the DBScan documentation, this could have been due to noise as well as us setting the Min parameter at 1, which was fairly low.
        ii. Secondly, we made an assumption that by separating the data set by year during initial clustering, we could overcome the potential for fire events occurring in the same area at two different times during the same year thereby causing a cluster to potentially represent two different fire events. Although we performed some comparison between the Emissions/NASA data within a formed cluster with unique fire event information from a separate data set (1.88m unique fires database from the USDA) and found many matches based on lat/long and span of time, there were also many instances where our created clusters spanned broad date ranges that could not be matched or were not representative of true events because they may have captured two different events in the same cluster. We began developing a method to iterate through the data to break apart clusters that spanned large date

ranges, but with a lower unique count of days represented but chose to end the process due to time constraints.

2. **Identifying Unique Fire Events** - As we continue to look into this project, we'd like to expand our ability to create cluster datasets that match the 1.88m unique fire dataset from the USDA. This would offer the ability to review each fire to determine if features could be created from the range of time for a sample fire (begin to end date), the change of size across dates using coordinate based expansion, movement in lat/long direction of the fire due to weather or land conditions, and use of information from the identification of unique fire events to expand on features as we learn more about the locational factors at play.
3. **Modeling Using Brightness Captured through Channel 31 of the MODIS** - As the iterations of the Software Engineering Development Lifecycle proceeded and the team revisited the choices of features and models, the question was raised whether Bright_T31 could've offered better quality data for our models. The team's answer is possibly yes. Bright_T31 does not have a skewed distribution and knowing that our models improved upon introducing synthetic data through SMOTE(), Bright_T31 might have helped the performance of our models.
    i. In other clusters, we found that although some date ranges matched exactly to a separate reference dataset we held that documented 1.88 unique fire events during our time range, other clusters held dates that expanded long time ranges that were clearly not related to a single event.

Appendix

1-A - Results from DBScan Clustering of 5.96m Emissions Records

Results from Clustering:

Emissions Data from USDA:

2003 - Clustered 317,353 points down to 6,231 clusters, for 98.0% compression in 44.70 seconds

Silhouette coefficient: 0.491

2004 - Clustered 164,634 points down to 6,568 clusters, for 96.0% compression in 21.16 seconds

Silhouette coefficient: 0.605

2005 - Clustered 399,828 points down to 9,010 clusters, for 97.7% compression in 76.10 seconds

Silhouette coefficient: 0.419

2006 West - Clustered 300,874 points down to 2,718 clusters, for 99.14% compression in 43.30 seconds

Silhouette coefficient: 0.359

2006 East - Clustered 190,156 points down to 9,073 clusters, for 95.2% compression in 27.19 seconds

Silhouette coefficient: 0.341

2007 - Clustered 640,039 points down to 8,945 clusters, for 98.6% compression in 110.69 seconds

Silhouette coefficient: 0.269

2008 - Clustered 440,888 points down to 9,264 clusters, for 97.9% compression in 75.88 seconds

Silhouette coefficient: 0.444

2009 - Clustered 375,841 points down to 8,234 clusters, for 97.8% compression in 57.06 seconds

Silhouette coefficient: 0.281

2010 - Clustered 270,486 points down to 8,237 clusters, for 97.0% compression in 30.41 seconds

Silhouette coefficient: 0.512

2011 - Clustered 770,255 points down to 10,554 clusters, for 98.6% compression in 137.69 seconds

Silhouette coefficient: 0.252

2012 - Clustered 667,029 points down to 7,549 clusters, for 98.9% compression in 114.98 seconds

Silhouette coefficient: 0.340

2013 - Clustered 272,406 points down to 5,483 clusters, for 98.0% compression in 34.88 seconds

Silhouette coefficient: 0.528

2014 - Clustered 442,103 points down to 8,059 clusters, for 98.2% compression in 75.23 seconds

Silhouette coefficient: 0.188

2015 - Clustered 483,703 points down to 8,588 clusters, for 98.2% compression in 83.23 seconds

Silhouette coefficient: 0.379

2-a - Jupyter Notebook & Python References

Section of WildFire Capstone Process	Notebooks	Optional Comments
Ingestion	DarkSkyWeatherPull.ipynb	Interface with DarkSky API to pull down weather information for lat/long and date/time.
Wrangling	DBScan_Clustering_Emissions.ipynb	Initial clustering of emissions records to create unique cluster areas to link record information between Emissions, NASA and Weather databases.
Wrangling	Assign_ClusterReference.ipynb	Merging cluster reference label determined through the DBScan to the Emissions records. Secondly, taking the cluster reference assignment results between the cluster centerpoints and the NASA M6 records in the Connect_Datasets_Clusters.ipynb notebook, and merging them into the NASA M6 original records for those matches within 150 euclidean distance.
Wrangling	Connect_Datasets_Clusters.ipynb	Performing K-D Tree analysis between the cluster centerpoints determined in the DBScan process, and the NASA M6 records by year, in order to assign a reference label to the NASA M6 Lat/Longs and
Wrangling	MLInputs.ipynb	Pulling together the Fire Intensity Model input file based on the Y target values in the NASA M6 Dataset, landscape features from the Emissions and Weather data.
Feature Creation	NASA_Regions	Using KMeans to create regions across the US based on lat/long for a new feature input.
EDA	NASA-StatisticalAnalysis-ProductionCode.ipynb	
EDA	Model-EDAandFeatureAnalysis-ProductionCode.ipynb	
EDA	Fire Intensity-Statistical Analysis-Production Code.ipynb	The result of extensive data wrangling and analysis oriented on a tailored fire intensity CSV file using statistical techniques learned in class
EDA	Emissions-Statistical Analysis-Production Code-Post.ipynb	The result of extensive data wrangling and analysis oriented on a tailored fire emissions CSV file using statistical techniques learned in class
EDA	ClusteringNASA-M6-ProductionCode.ipynb	
EDA	ConnectingClusterPoints-NASAandUSDA-ProductionCode.ipynb	
Model	Model_Hyperparameter_Tuning.ipynb	
Model	Initial_Feature_And_Model_Evaluation.ipynb	
Model	Final_Model_Evaluation.ipynb	

Product	FireSeverityApp.ipynb	
Product	Firepredict.py	This Python file runs the app. Need to make sure the config.py file and the pipeline.pickle file are in same folder

References

US Department of Agriculture - Missoula Fire Lab Emission Inventory (MFLEI) for CONUS
https://www.fs.usda.gov/rds/archive/catalog/RDS-2017-0039

NASA -
https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms/c6-mcd14dl#ed-firms-attributes

Machine Learning Application in Wildfire Science and Management:
https://www.groundai.com/project/a-review-of-machine-learning-applications-in-wildfire-science-and-management/

How to predict the spread and intensity of forest and range fires:
https://www.fs.fed.us/rm/pubs_int/int_gtr143.pdf

Fire behavior prediction and fuel modeling system: https://www.fs.fed.us/rm/pubs_int/int_gtr260.pdf

Fire intensity, fire severity and burn severity: A brief review and 1 suggested usage - Jon E. Keeley:

https://www.firescience.gov/projects/04-1-2-01/project/04-1-2-01_04-1-2-01_ms_aafireintensityseverity_nov2007_complete.pdf

Clustering Data Points using DBScan: Vibhor Argowal

https://medium.com/@agarwalvibhor84/lets-cluster-data-points-using-dbscan-278c5459bee5

Tools used in support: Slack, Trello, DBDesigner.net, SiteGround, Google Drive, MySQL Relational Database Management System (RDMS)