Applied Competitive Lab in Data Science – Final Project

Predicting causes of wildfires in the USA

Lecturer: Yonatan Schifter

Work by Group 8:

Raz Bareli - 203488747 Shir Levi - 316419423

Yuval Omer – 316514314 Maya Swissa - 209028760

Introduction - Predicting causes of wildfires in the USA

With almost a million and a half fires, more than three thousand deaths, and close to 22 billion dollars of damage reported in the USA in 2020 alone - fire detection and prevention is imperative for public safety.

Fire investigation is the analysis of fire-related incidents through the use of fire dynamics. When a disastrous fire occurs, investigators must determine where and how the fire started, and whether it was accidental or intentional. If an intentional fire was set, the investigator's findings could lead to criminal charges.
Like any other investigation, time is of the essence when investigating fire cause. As such, creating ML methods to assist investigators in narrowing in on plausible fire causes is a point of interest in today's fire investigator teams.

"Using AI to determine the cause of fires can be a valuable tool for investigators, as it can help them to identify patterns and potential causes more quickly and accurately. By analyzing large amounts of data from multiple sources, AI can help to uncover hidden patterns and relationships that may not be immediately apparent to human investigators." (chatGPT when asked about AI and fire cause investigation).

With even the greatest mind of our time, chatGPT, expressing such passion in developing ML powered AI for fire cause investigation - it is our duty as hatchling data scientists to try our hand at developing ML powered fire cause prediction models.

In this project report we have broken down the model development into four stages:

1. Exploring and Cleaning Data
2. Feature Engineering
3. Tuning and Evaluating
4. Feature importance and outlier detection

In each part we will present our work done in accordance with the work stage and discuss certain aspects of our work, including methods that worked and methods that didn't.

Afterwards we will discuss our conclusions from the project, where and how we believe the project can be improved upon and future avenues for research into better ML powered fire cause predictors.

Part 1: Exploring and Cleaning Data:

In this part we'll go over the data that we received. We'll talk about ways of visualizing and exploring the data so we can gain a better understanding of it.

Filtering Features:

This Is a list of all the features in the data set:

```
OBJECTID', 'FOD_ID', 'FPA_ID', 'SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM',
    'NWCG_REPORTING_AGENCY', 'NWCG_REPORTING_UNIT_ID',
    'NWCG_REPORTING_UNIT_NAME', 'SOURCE_REPORTING_UNIT',
    'SOURCE_REPORTING_UNIT_NAME', 'LOCAL_FIRE_REPORT_ID',
    'LOCAL_INCIDENT_ID', 'FIRE_CODE', 'FIRE_NAME',
    'ICS_209_INCIDENT_NUMBER', 'ICS_209_NAME', 'MTBS_ID',
'MTBS_FIRE_NAME',
    'COMPLEX_NAME', 'FIRE_YEAR', 'DISCOVERY_DATE', 'DISCOVERY_DOY',
    'DISCOVERY_TIME', 'STAT_CAUSE_CODE', 'STAT_CAUSE_DESCR', 'CONT_DATE',
    'CONT_DOY', 'CONT_TIME', 'FIRE_SIZE', 'FIRE_SIZE_CLASS', 'LATITUDE',
    'LONGITUDE', 'OWNER_CODE', 'OWNER_DESCR', 'STATE', 'COUNTY',
    'FIPS_CODE', 'FIPS_NAME', 'Shape'
```

Before starting to explore the data in depth we first need to determine which features we can use, and which features risk leaking information on the label of the sample.

The unusable group of features are:
FOD_ID, FPA_ID, MTBS_ID, SOURCE_SYSTEM_TYPE, SOURCE_SYSTEM, NWCGREPORTINGAGENCY, NWCGREPORTINGUNIT ID, NWCGREPORTINGUNIT NAME, SOURCEREPORTINGUNIT NAME, LOCALFIREREPORT ID, LOCALINCIDENTID, FIRE CODE, FIRE NAME, ICS209INCIDENT NUMBER, ICS209NAME, MTBS ID, MTBSFIRENAME, COMPLEX NAME, OWNER CODE, OWNER DESCR, FIPS CODE, FIPS NAME, shape.
All those contains several types of ID’s or names representing the sample in multiple datasets or fire fighter units, these features are at best irrelevant to the prediction of the fire or at worst can cause data leakage if some datasets only store information of fires started by a certain cause e.g. Arson fire database.

The usable group of features:
The Features we will explore more in depth to extract more features from and use for training, we’ll look at our individual features' values and their distributions.

	FIRE_YEAR	DISCOVERY_DATE	DISCOVERY_DOY	STAT_CAUSE_CODE	CONT_DATE	CONT_DOY	FIRE_SIZE	LATITUDE	LONGITUDE	OWNER_CODE
count	626822.000000	6.268220e+05	626822.000000	626822.000000	3.297340e+05	329734.000000	626822.000000	626822.000000	626822.000000	626822.000000
mean	2003.707373	2.453063e+06	164.719997	5.974548	2.453240e+06	172.799924	70.006015	36.779002	-95.724077	10.599197
std	6.659931	2.433510e+03	89.995807	3.484163	2.684869e+03	84.243639	2169.517330	6.141014	16.726271	4.402621
min	1992.000000	2.448622e+06	1.000000	1.000000	2.448622e+06	1.000000	0.000010	17.939722	-178.802600	0.000000
25%	1998.000000	2.451086e+06	89.000000	3.000000	2.450708e+06	103.000000	0.100000	32.815319	-110.427737	8.000000
50%	2004.000000	2.453174e+06	164.000000	5.000000	2.453470e+06	181.000000	1.000000	35.455000	-92.079745	14.000000
75%	2009.000000	2.455036e+06	230.000000	9.000000	2.455750e+06	232.000000	3.270000	40.831903	-82.300308	14.000000
max	2015.000000	2.457388e+06	366.000000	13.000000	2.457388e+06	366.000000	537627.000000	70.138100	-65.264175	15.000000

We'll devide out features into groups, just for the sake of order:

Ordinal features:
Fire year – the year the fire was discovered in.
Discovery Day of year – the day of the year (0-365)
Discovery Time – hour of discovery
Cont Day of year – the day of the year (0-365)
Cont Time – hour of cont.

Continues features:
Discovery date – the date the fire was discovered.
Cont date – the date the fire was under control.
Fire size – Estimate of acres within the final perimeter of the fire.
Latitude, Longitude – geographical location where the fire was discovered.

Categorial features in the dataset:
Fire size class – A categorical version of the Fire size data.
State– the state in which the fire was discovered.
County – in which the fire was discovered.

<u>Exploring null values:</u>

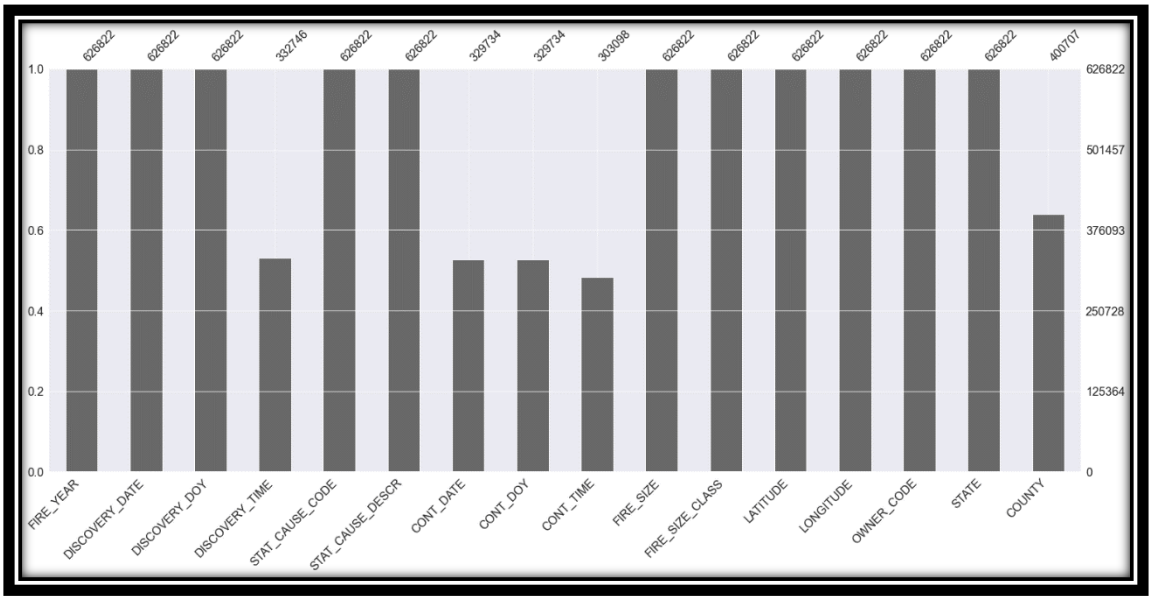
We explored all null values in our data set:

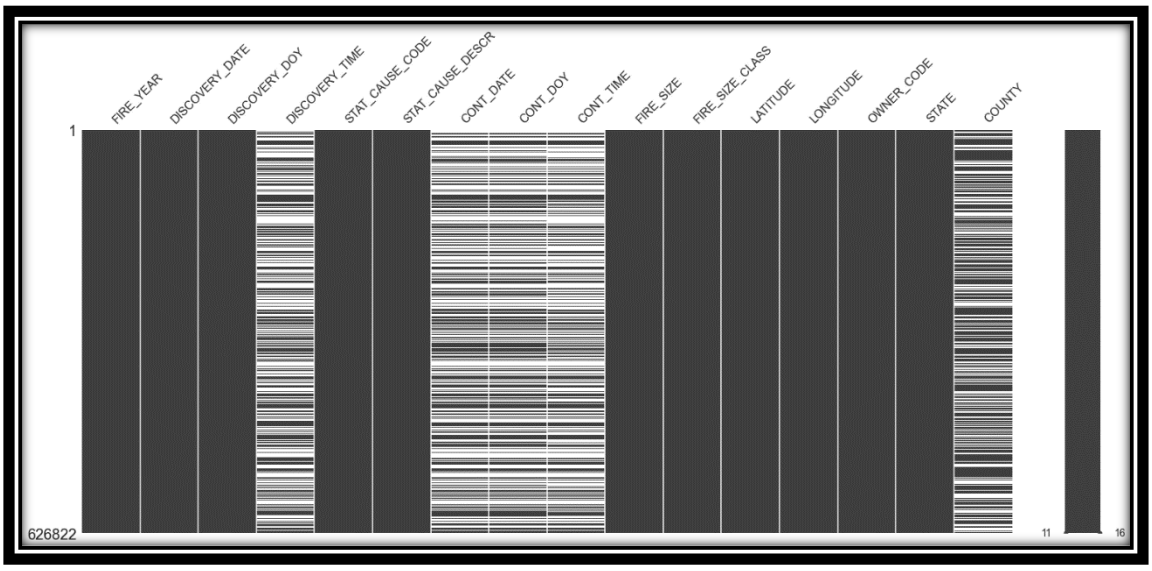
```
FIRE_YEAR              0
DISCOVERY_DATE         0
DISCOVERY_DOY          0
DISCOVERY_TIME    294076
STAT_CAUSE_CODE        0
STAT_CAUSE_DESCR       0
CONT_DATE         297088
CONT_DOY          297088
CONT_TIME         323724
FIRE_SIZE              0
FIRE_SIZE_CLASS        0
LATITUDE               0
LONGITUDE              0
OWNER_CODE             0
STATE                  0
COUNTY            226115
dtype: int64
```

We see that large amount of the data has no Discovery Time, Cont date, Cont Day of year, Cont Time values. As we learned, data points containing null values can either be discarded or imputed. While it could be a bit troublesome to impute the duration

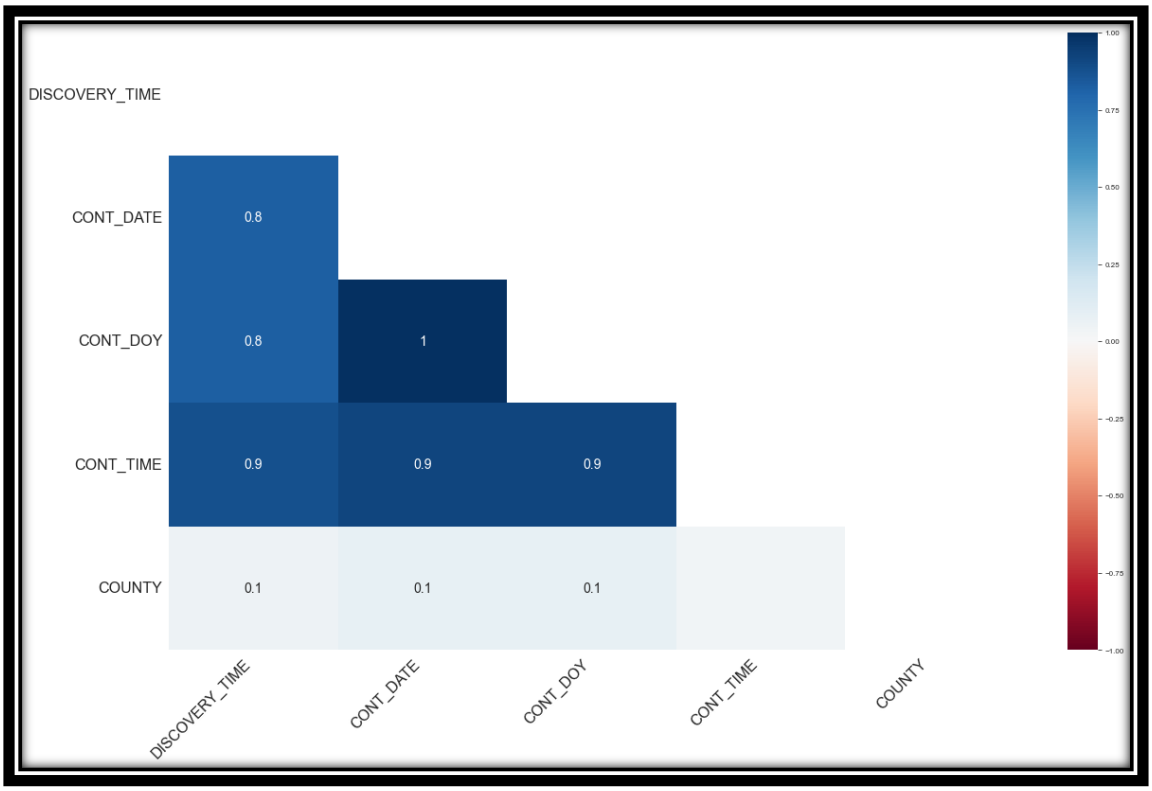
of the fire, for example, we'll try to use the other features and make the best of it. We will first approach the problem 'head on' with no prior assumptions.

We used the missingno package to visualize missing data:





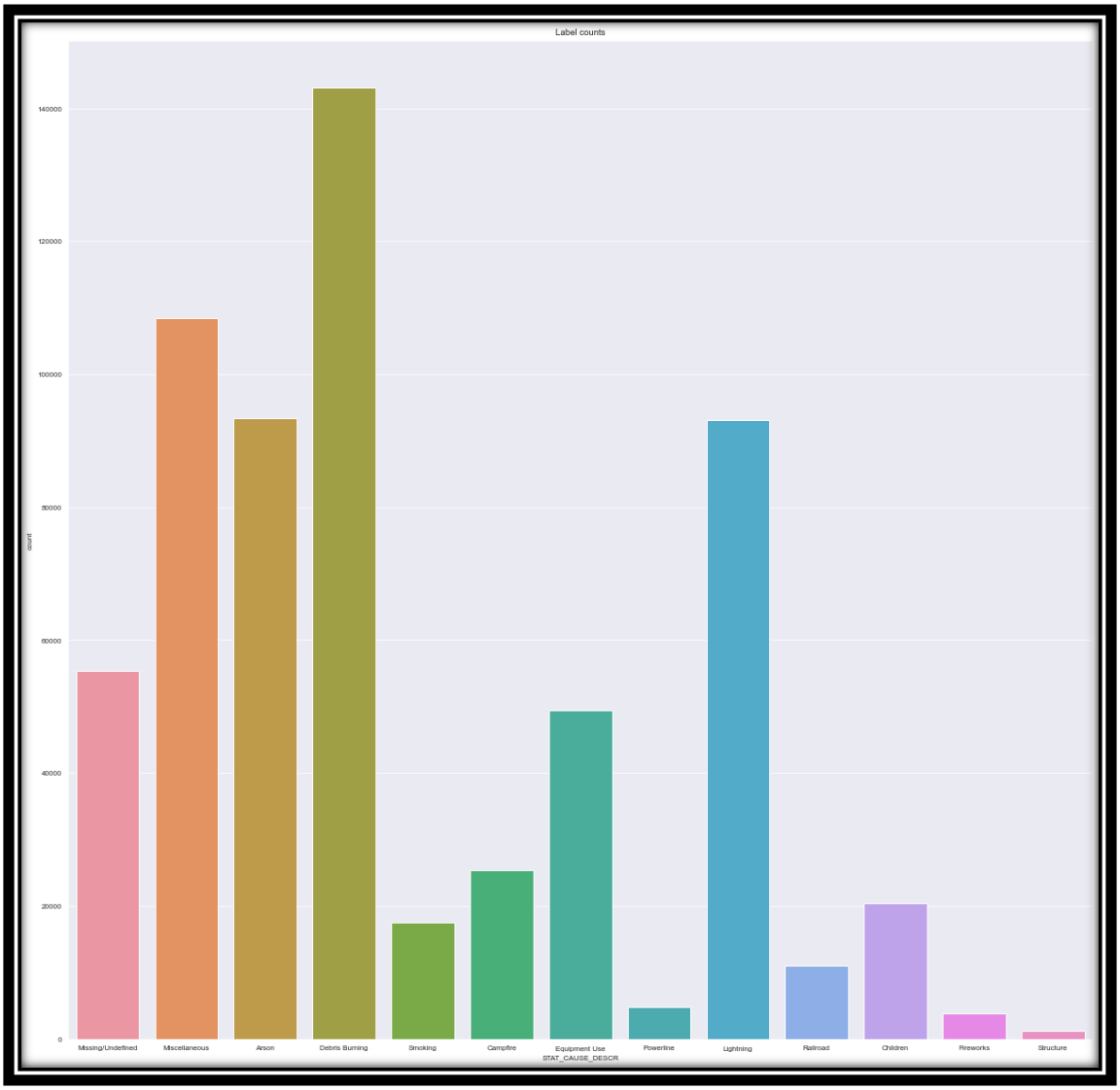
As we already noticed, there are missing values at columns:
DISCOVERY_TIME, CONT_DATE, CONT_DOY, CONT_TIME, COUNTY.
Moreover, it is seeming from the matrix that CONT_DATE and CONT_DOY had null values in the same rows, and also that there is great similarity with CONT_TIME and DISCOVERY_TIME as well. We'll investigate it with the following heatmap:



This heatmap shows us correlation between the features in terms of null values. We can see that, as we expected, CONT_DATE and CONT_DOY have null values for the same samples. Also, as we expected, we notice that there is great correlation also with CONT_TIME and DISCOVERY_TIME. On the other hand, we see that there is no significant correlation with the COUNTY feature.

Exploring and analyzing the data, visually:

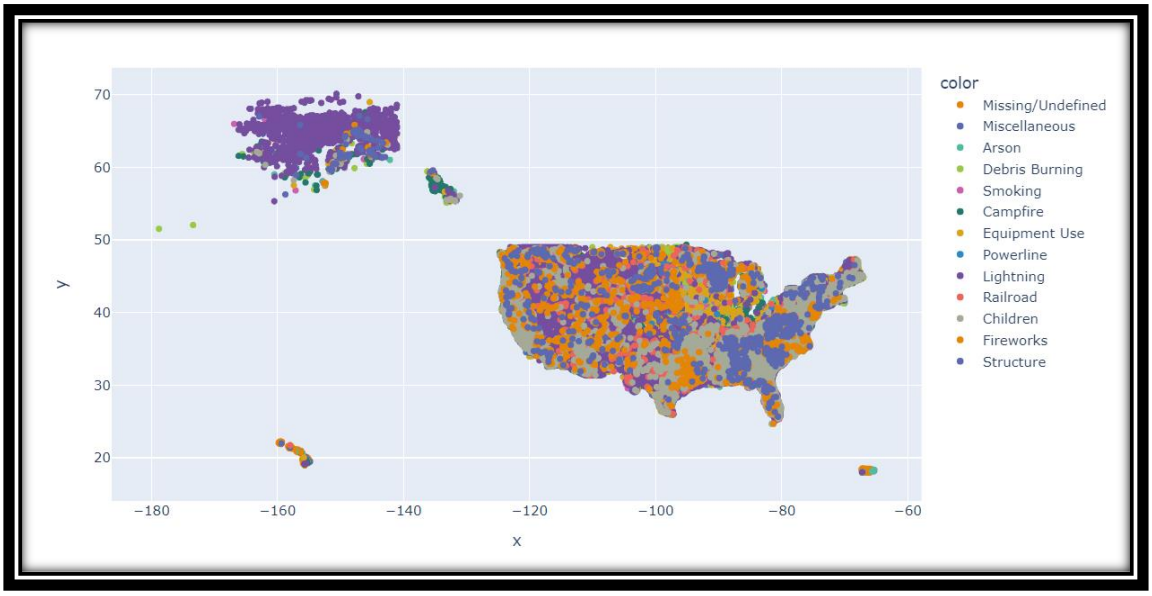
First, we wanted to see all the causes and have an idea of what causes most of the fires in the US, and what are the least common causes. For that purpose, a count plot is very suitable.



This plot explains the causes in a visual way that enables us to understand the occurrences of fires from the different causes. As we can see, some causes are very common in USA, such as Derbies Burning and Miscellaneous, while others are very rare, such as Structure or Powerline.

We started out with several hypothesis about your features and try to assert how true they are. We checked the connection between the fire causes and its coordinates, it's size and the year when it occurred.

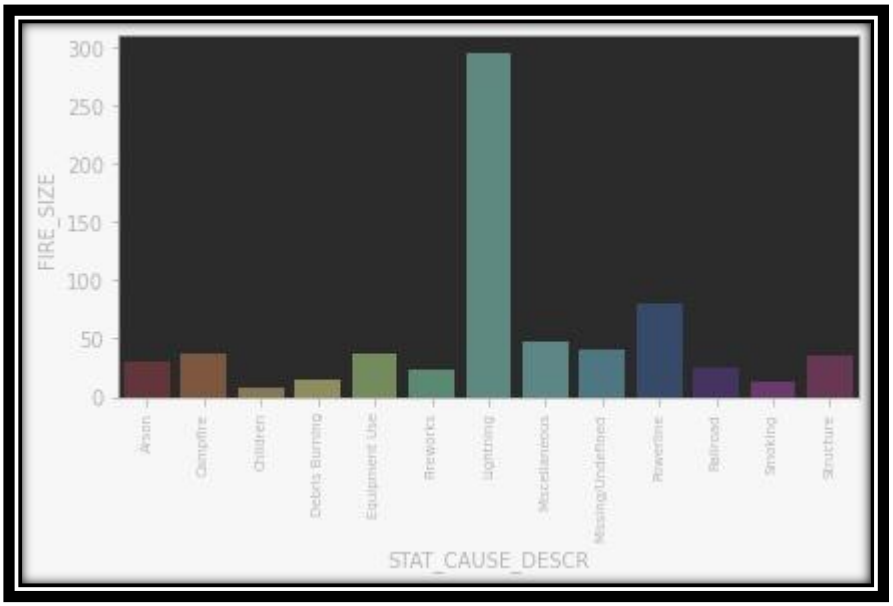
First, we wanted to have a visual look at the coordinates of the fire and see if we can get some information out of it:



This is a plot of the Lat and Long values of our training data, with color corresponding to our label, meaning what caused the fire.
We can learn a few things from this plot:
The first being that all the latitude and longitude values in the data are in the legal range and are corresponding to real locations in the territories of the USA.

We will notice that there are a few points that seem to be outside of the USA, but upon conferring with maps we have seen that those positions correspond to: Puerto Rico, Hawaii and islands off the coast of Alaska.

Second, we wanted to see if there's anything we can learn about the causes by the size of fires caused by them:



The above graph is representing the average fire size per cause of fire.
From this we can see that lightning has by far the largest average fire size which means Fire size might be an important feature for predicting Lightning as a fire cause.

Lastly, we explored the connection between the fire year (when it occurred) and the causes. We were able to notice some interesting facts:

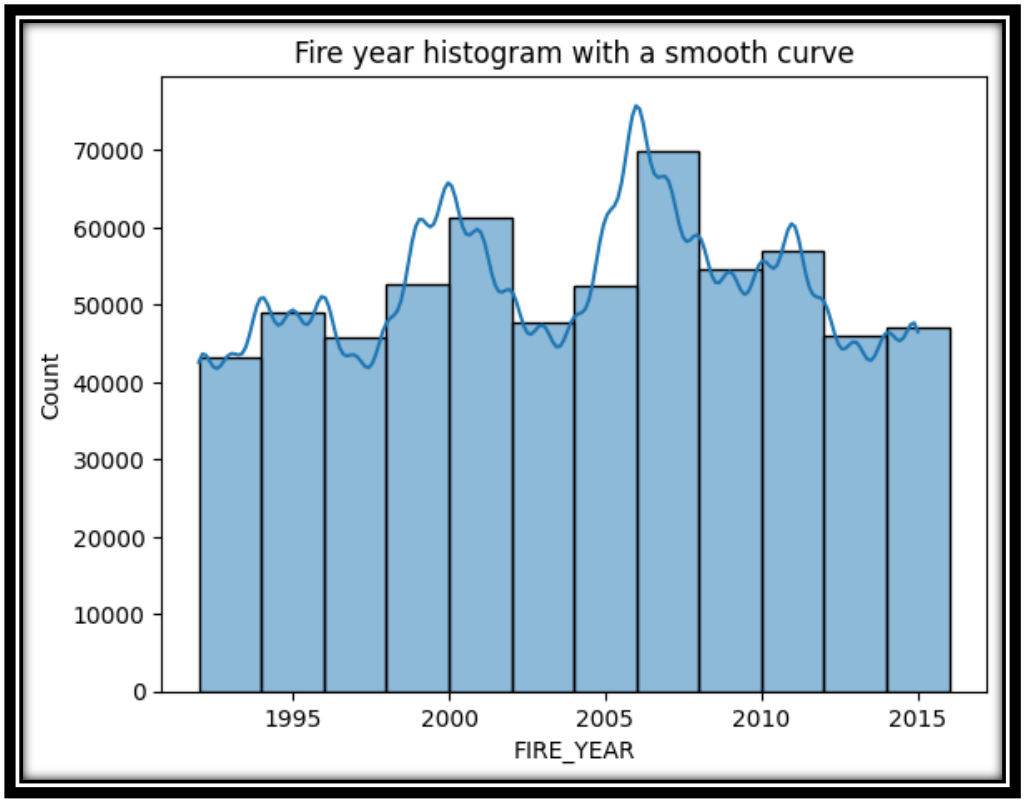
STAT_CAUSE_DESCR						
Arson	3688	3671	5948	3133	2263	2180
Campfire	895	907	910	1114	1262	1197
Children	923	864	1186	745	491	467
Debris Burning	4260	4122	7354	6327	6522	6034
Equipment Use	1882	1846	2182	2512	1530	1819
Fireworks	69	70	96	158	195	183
Lightning	4071	2492	5531	3298	2835	3335
Miscellaneous	2500	2281	6520	6975	4906	6069
Missing/Undefined	2735	2905	474	3292	1221	2093
Powerline	35	34	112	98	490	501
Railroad	688	611	1075	246	177	123
Smoking	883	840	996	585	421	651
Structure	13	19	42	32	85	106

At first, we started looking at the rates of the different causes by the year (we sampled some years to give the intuition: 1992, 1993, 2000, 2008, 2014, 2015). As we see, the rates of the fire cause, in some cases, stays pretty much the same, as in Structure. On the other hand, we see that, in some cases, the rates of the causes change during the years. They might increase during the years, as we see in fire

caused by Powerlines, or decrease, as we see in Railroad fires. There are also cases where the behavior is not stable, as we can see in Equipment Use.

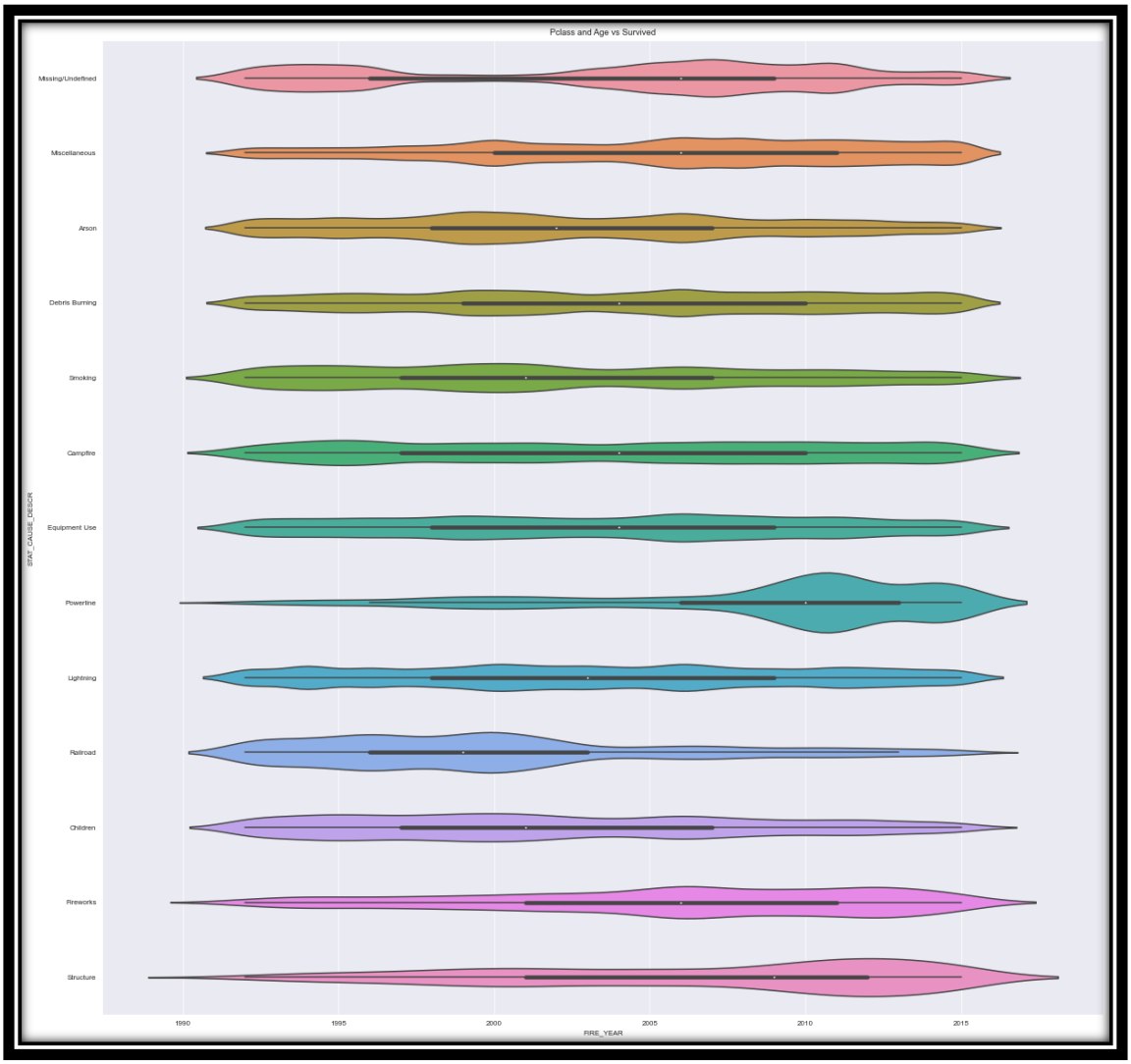
As we can see, there are no patterns, but we can only see that in some years more fires occurred than in others. We investigated it to try and see in a more visual way the rates of different causes per year:

We wanted to see if any year had some more fires in the following plot:



We see that year 2006 had the highest number of fires, and we can see that there were peaks as well around years 2000 and 2011.

We looked for a better way to see the connection with the causes as well.



We learn from this violin plot how was the spread of one cause over the years. As we saw in the table, there is an increase during the years of fire caused by Powerlines, as well as Fireworks and Structure. There is a decrease in Railroad fires, as we noticed in the table before, and as well in fires caused by Children.

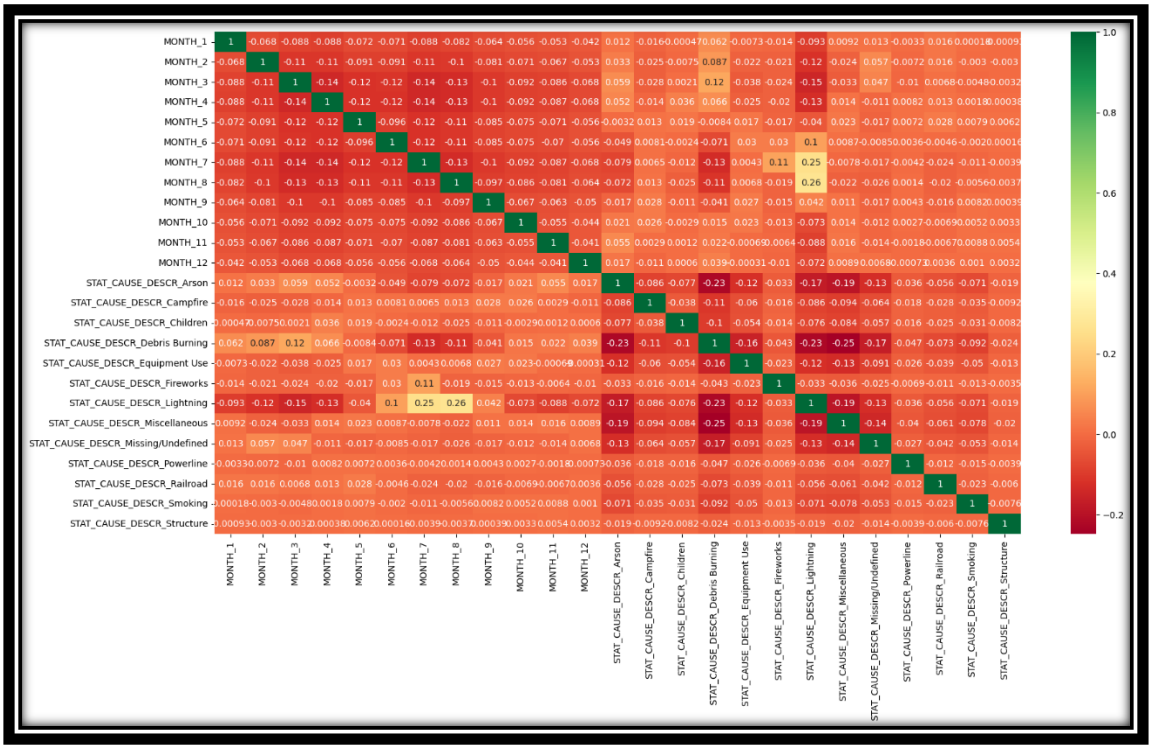
Preprocessing: clean usable data

we preprocessed all features to create clean usable data for the model.
We first reformed all dates and hours that will enable computing with it. We added the columns: DATE, DAY, MONTH, YEAR, C_DATE, C_DATE_EXISTS (as we saw, there are Na values), EXISTS_TIMES (as we saw, there are Na values), HOUR, C_HOUR, DURATION, DURATION_DAYS, DURATION_HOURS.

We created DATE and C_DATE in format of datetime64[ns] that would enable us do calculates. As well we took some values out of it to check for correlations: DAY, MONTH, YEAR (of discovery). We added HOUR and C_HOUR, when available, for calculating the duration of fire. Also – we added the duration in timedelta form, and added features for most seen values in order to group it in an organized way.

As we saw before, the years might matter when it comes to deciding the cause, and so we figured that the dates, hours, and duration of the fire might be critical as well. The only problem was that this data is not well organized, and not easy to use and find some similarities. Therefore, we calculated and created the features mentioned above, to try and look for more common features.

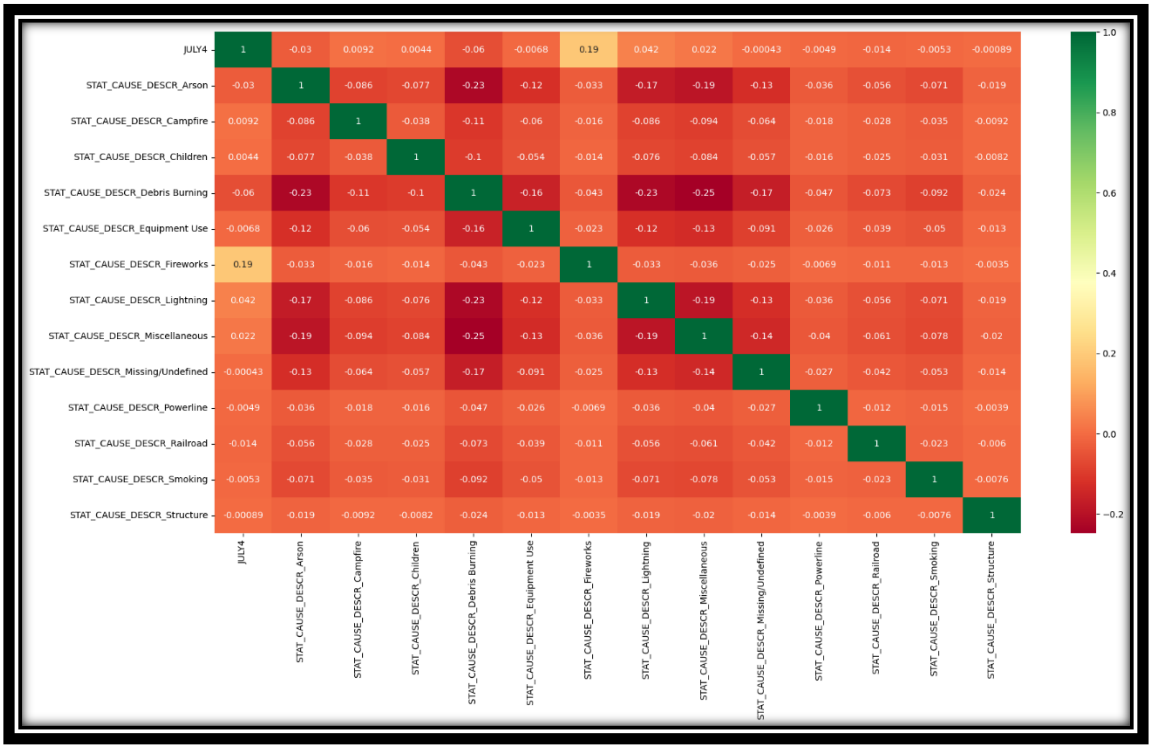
We explored our new features' correlation using a heatmaps, to see if we're onto something:



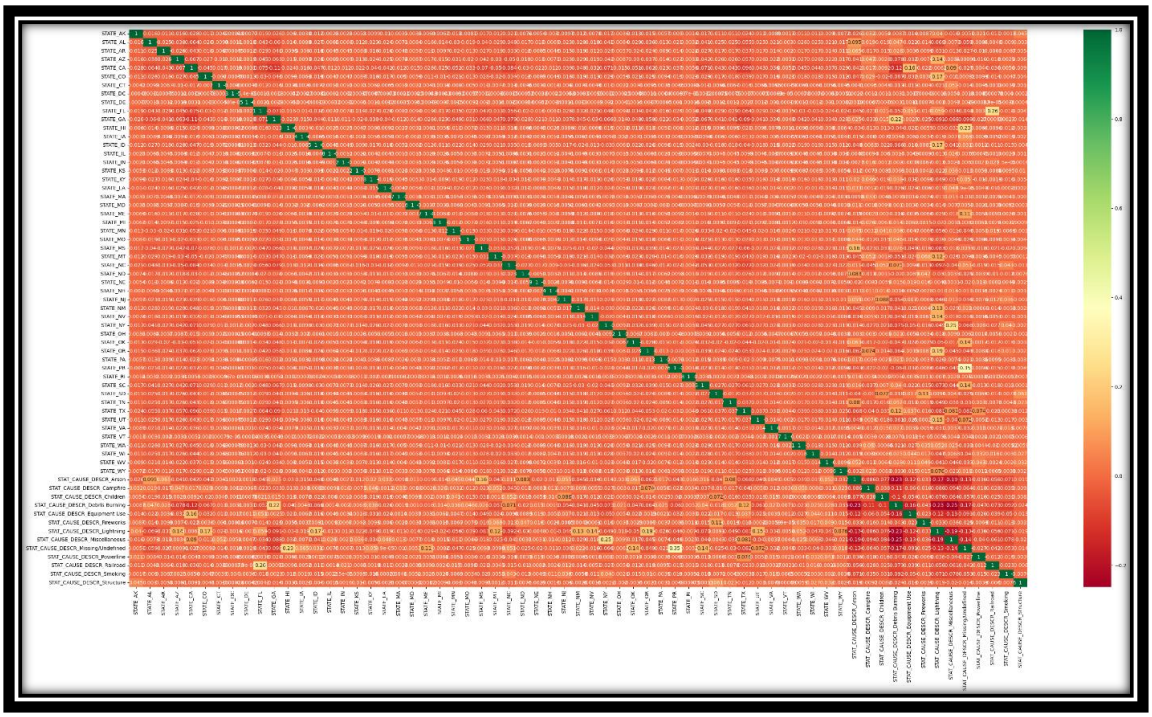
The above graph is a covariance matrix between fire causes and months, from

It we can learn the following: (1) Lightning as a fire cause is more likely to occur between June to august, which makes sense since those months are Hurricane season in some states. (2) Debris Burning are correlated with months February and March. (3) We see that there is a correlation between July and Fireworks, and that makes since as July 4th is very much associated with fireworks.

We'll investigate it to ensure our suspects:



The above graph is a covariance matrix between July 4th and fire causes, ensuring our suspects. We can see that fireworks has a value of 0.19 for the covariance value with July 4th. Comparing this value with the covariance value of fireworks with the moth of July (0.11), we can deduce that fireworks as a fire cause is especially likely on July 4th.

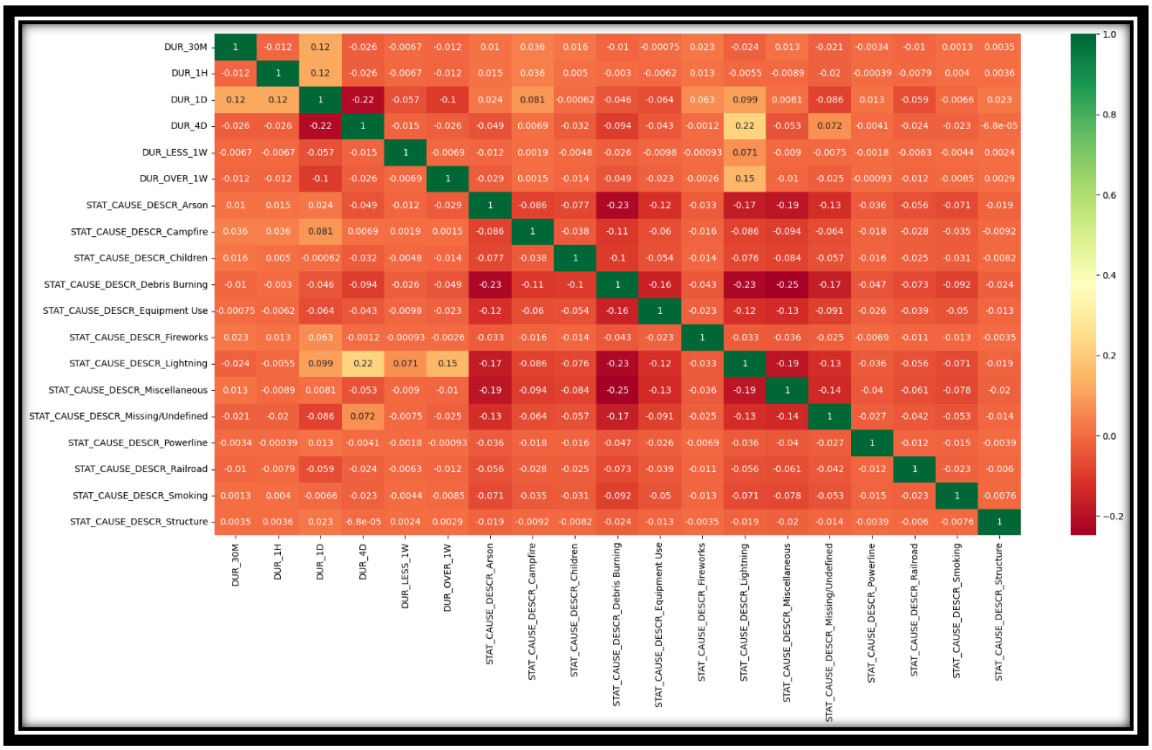


The above graph is a covariance Matrix between fire causes and states in the USA. From this matrix we can learn the following:

(a) some states have a higher correlation with some type of fire cause.

(b) Some causes have a higher correlation with states that have a lower population for e.g.: Lightning.

(c) some states (Hawaii, Puerto Rico…) have great correlation with cause missing/undefined.

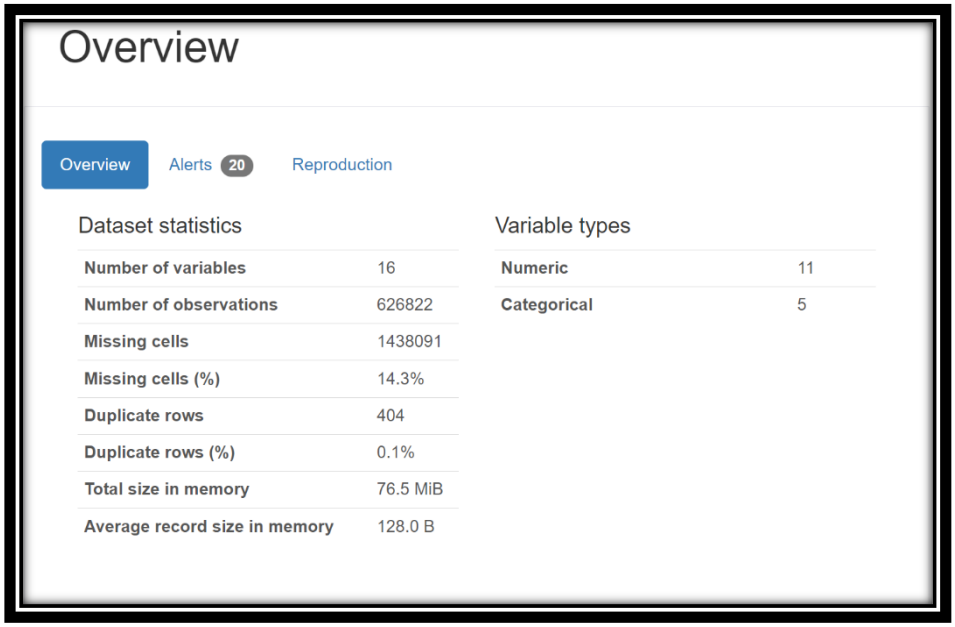


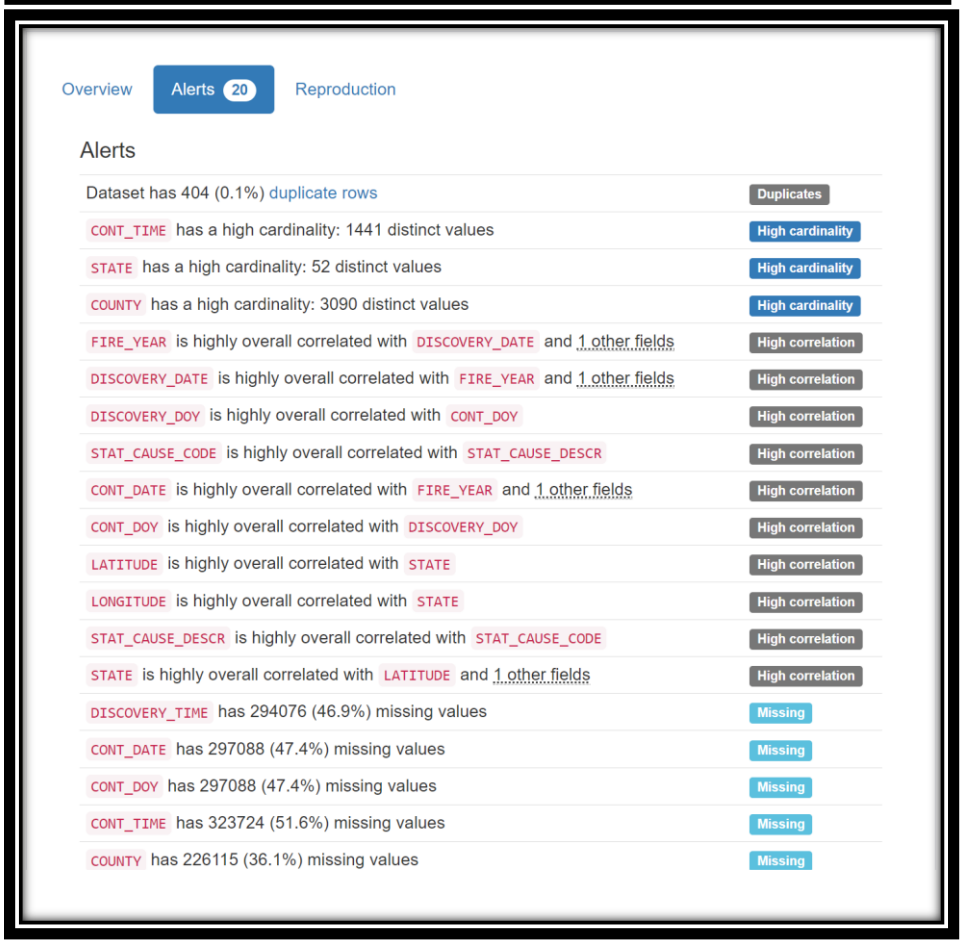
The above graph is a covariance matrix between time passing until the fire is under control and fire causes. We can see that (1) Campfires have some correlation with up to 1 day handling, and some correlation with 30 minutes and 1 hour fire. (2) There is some correlation between Fireworks and fires ending in same day. (3) Fires caused by Lightning have great correlation with lasting over 1 day (up to 4 days or over a week).

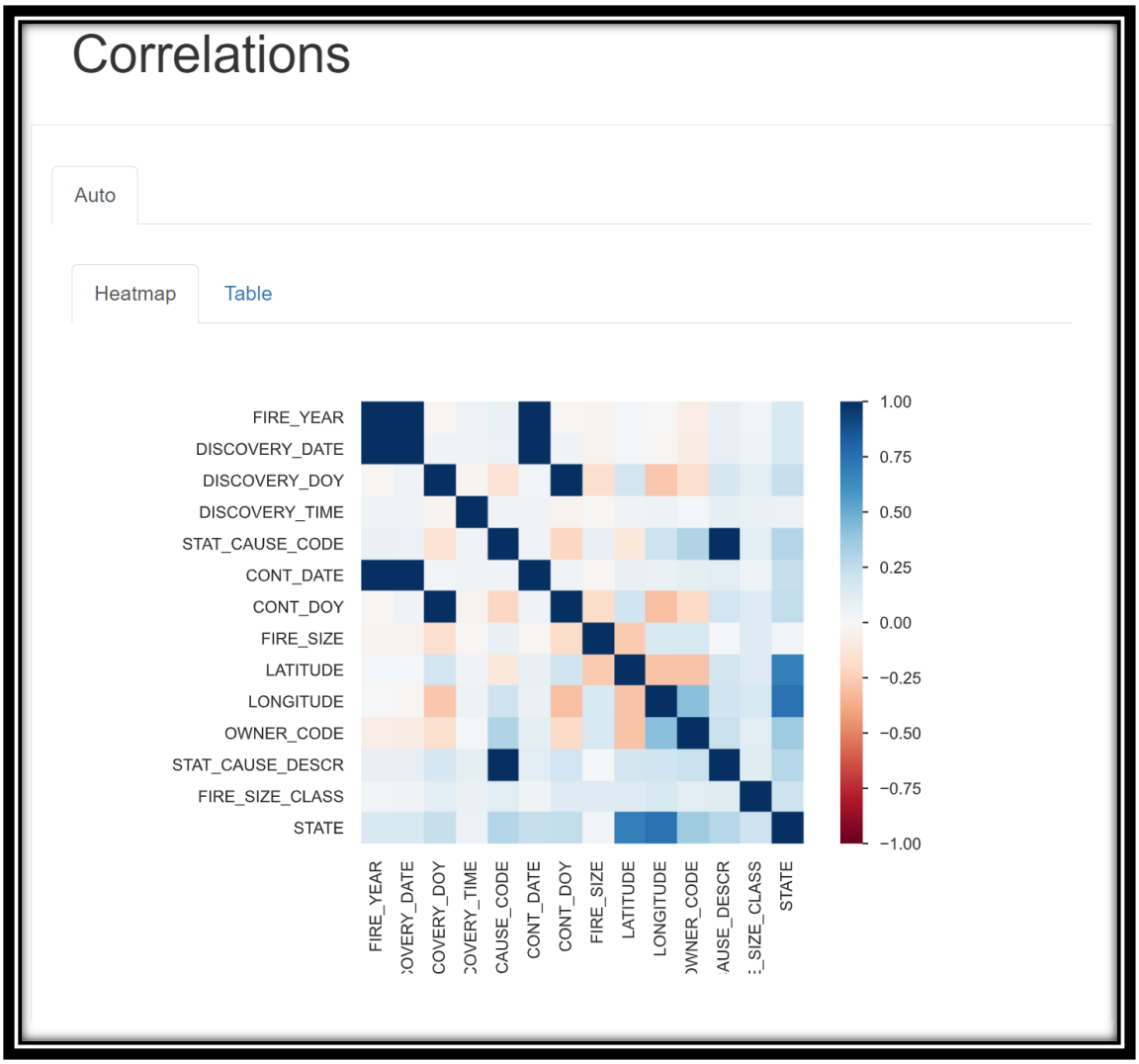
As we see, using the heatmap and exploring our features' correlation, the months, states, and duration time of the fire might be correlated with the cause of the fire. To explore the data, we added some more features such as JULY4, DUR_30M, DUR_1D etc.

We used all new features to continue analyzing the data.

Python Overview: we use the pandas profiling library to get more ideas to explore.







From this heatmap, we see that there is a great correlation between the discovery day and the cont day – which have missing values. We wanted to explore it once more.

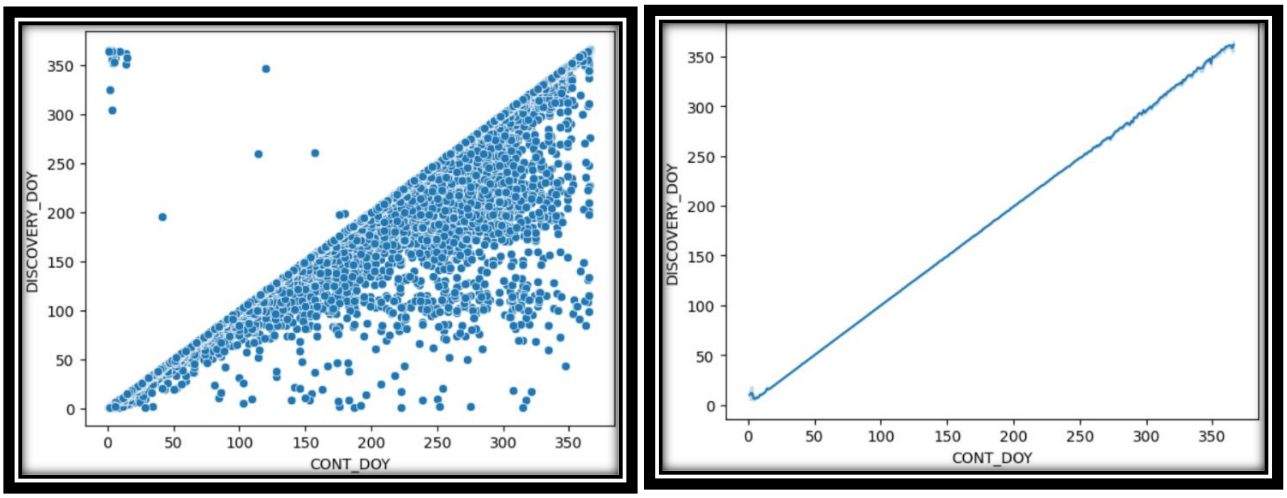
<u>Handling missing values:</u>

```
FIRE_YEAR              0
DISCOVERY_DATE         0
DISCOVERY_DOY          0
DISCOVERY_TIME    294076
STAT_CAUSE_CODE        0
STAT_CAUSE_DESCR       0
CONT_DATE         297088
CONT_DOY          297088
CONT_TIME         323724
FIRE_SIZE              0
FIRE_SIZE_CLASS        0
LATITUDE               0
LONGITUDE              0
OWNER_CODE             0
STATE                  0
COUNTY            226115
dtype: int64
```

As we saw already, there are many missing values. It mostly creates a problem when trying to calculate the duration of the fire, as we see that all the discovery dates are available, but many cont* dates are missing.

We'll look if we can replace the null values with more meaningful values to improve the prediction.

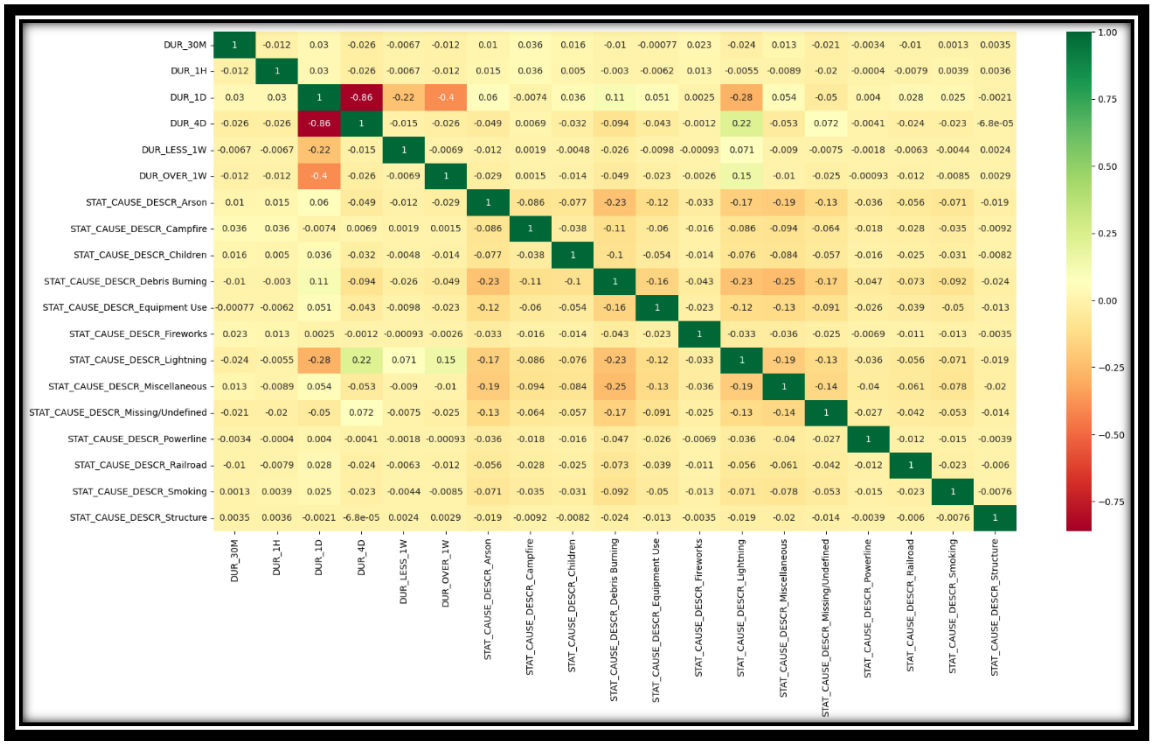
On the last heatmap, we saw there is correlation between the date the fire started and the date when it ended. We wanted to look if that might mean that replacing the missing value with the date of the discovery would be better. To do so we'll use the following plots:



As we see in the right plot, it seems like the correlation is strong. On the other hand, the other graph tells a different story: indeed, it seems like many fires starts and ends in the same day, but not necessarily always. From the left plot we learn much important things:

(1) Almost all cont dates are same or greater than the discovery date.
(2) Some cont dates are earlier than the discovery date, and that doesn't make any sense. But, if we notice, we see that it is earlier in up to ~365 days. From that we understood the year might not been updated. So – for every value of existing cont date, we checked if it is earlier than the discovery date, and if so, we added one year to fix it. That seemed to solve the issue.
(3) We understand there is no certainty that a fire with missing values for the cont date are the same day of the discovery.

We checked if changing the date from a default date to the discovery date makes difference.



We see that it has much in common with the correlations we saw using the default values.

Part 2: Feature Engineering

In this part we will focus on methods for expanding our set of features, and also modify the existing features that we got with the data. We believe that this is the most crucial part in this project, so we spent a lot of time thinking about how we want to modify the given features, and what features will help us with completing the specific task we got.

Adding features to dataset:

For starters since our dataset is relatively thin on meaningful features, we have decided to use our geospatial and date features to expand our dataset.

The features we have added are:

1. population density at fire location:
2. distance to nearest railroad:
3. distance to nearest powerline:
4. distance to nearest landfill:
5. distance to nearest public school:
6. distance to nearest home-park: RV and stuff
7. distance to nearest city (and name of that city):
8. info from nearest weather station: 1 to 3 days before the fire, the data is: temp, max min and mean, evaporation, precipitation
9. rate of smoking in % of population in each state in the year of the fire :
10. % of citizenships per state
11. races in state in %
12. demographics (ages)

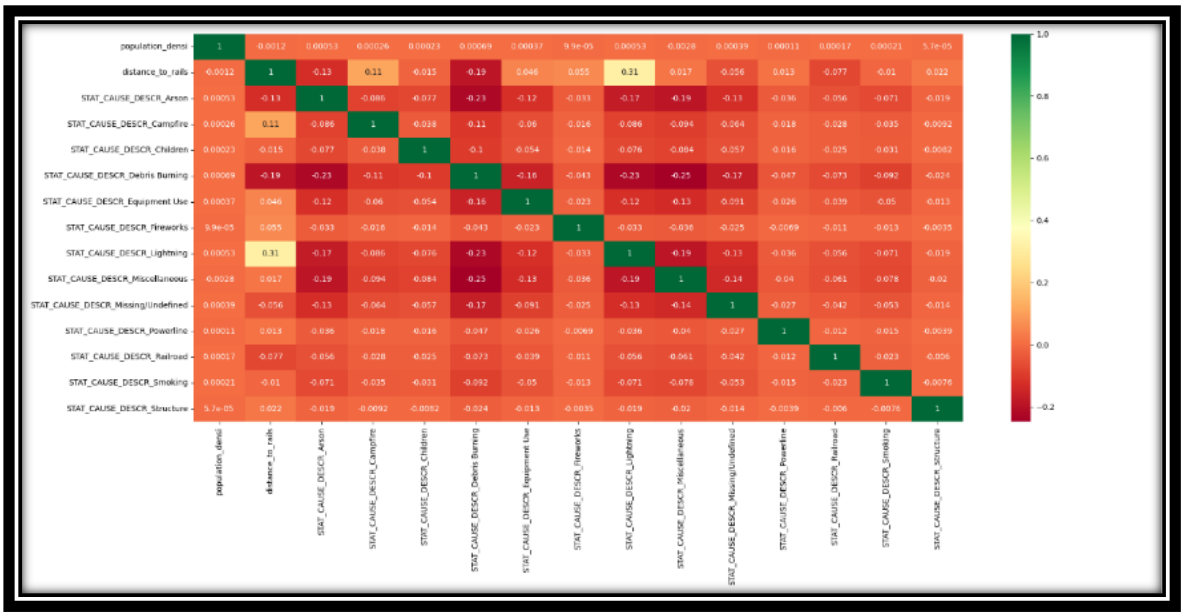
Population density at fire location:

Since many of the fire causes are either human caused or human adjacent (e.g. Fireworks, smoking, structure) we believe the population density can be important in determining fire cause. To get the population density of each sample in our dataset we used the SEDAC gridded population of the world v4 and used geopandas join_nearest to get the density population at the location of the fire discovery.

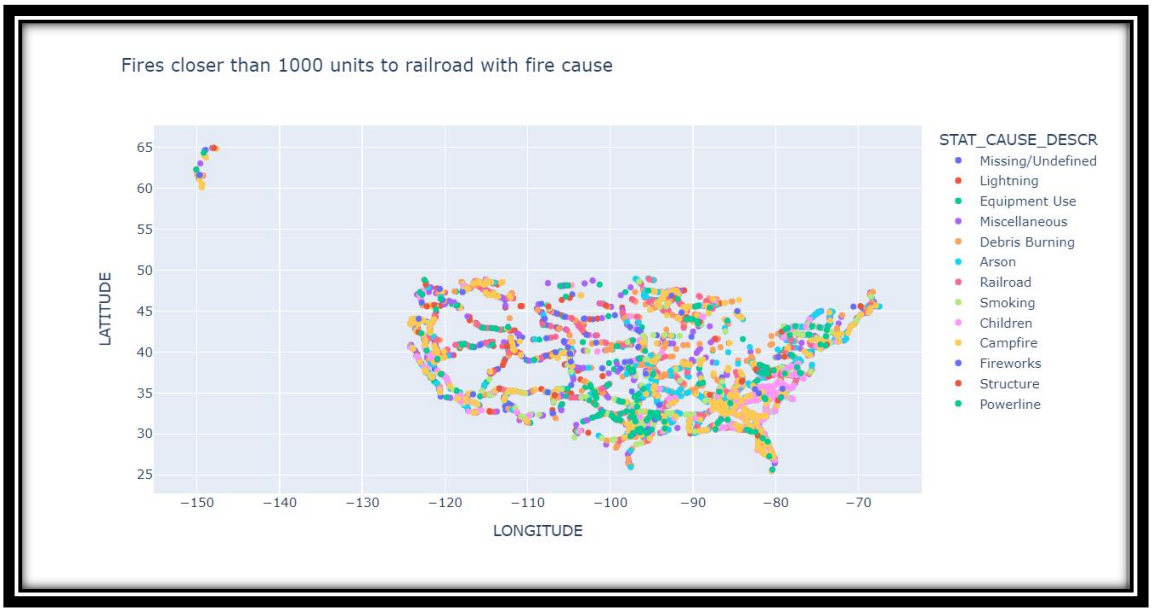
Distance to nearest railroad:

Since one of the fire cause descriptions is “railroad” we added the distance between the fire discovery location and the closest railroad. To achieve this we used a shapefile containing all the railroads in the US and calculated distance from each sample to the closest railroad.

Below we can see the covariance matrix between distance to nearest railroad and the fire cause descriptor.



Here we can see a plot with all fires that occurred close to railroads:

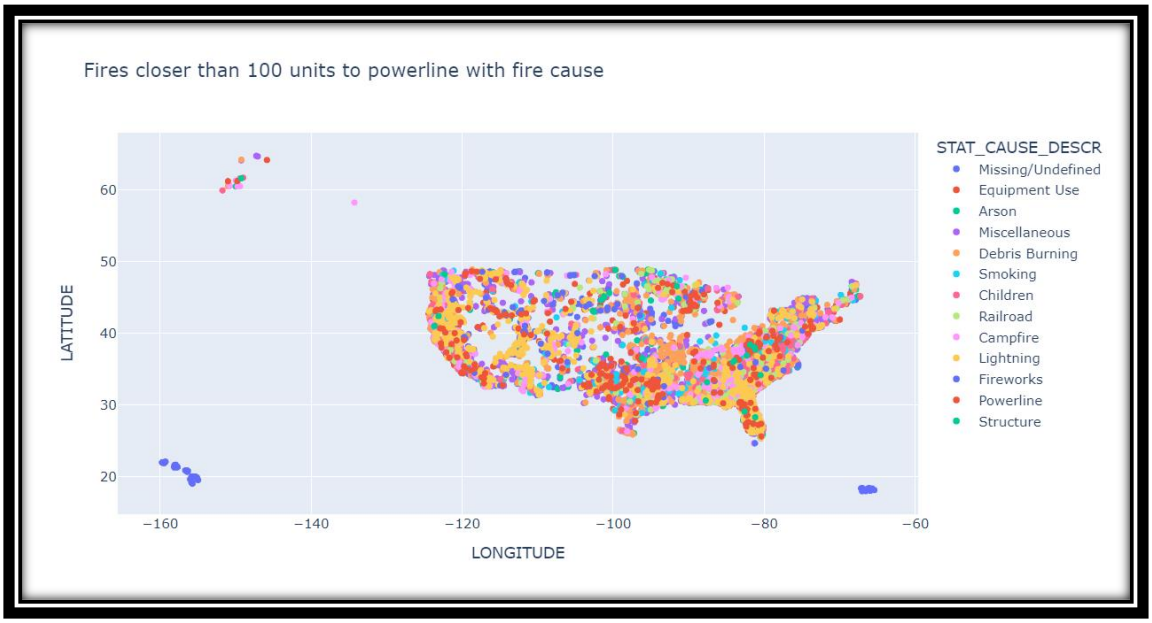


We can see in the graph above that while many fires originated next to the railroad systems, many of those fires are not classified as the railroad fire cause. Which means any model using this feature will need to rely on more features to reach a classification for the railroad fire cause.

<u>Distance to nearest Powerline:</u>

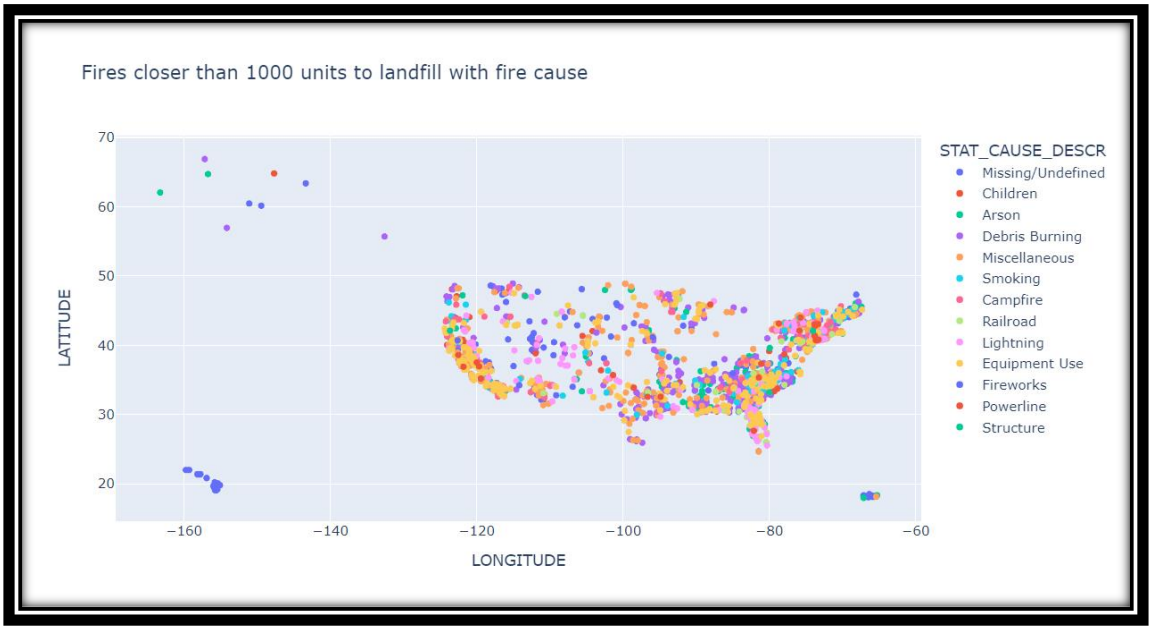
Since one of the fire cause descriptions is “Powerline” we added the distance between the fire discovery location and the closest railroad. To achieve this we used

a shapefile containing all the Powerlines in North America and calculated the distance from each sample to the closest Powerline.



Distance to nearest Landfills:

According to the U.S Fire administration an average of 8,300 fires have their starting location in a landfill and that 40% of those fires are attributed to arson.
To address this detail we have added a feature that contains the distance between each sample and the closest landfill location.



Distance to nearest Public school:

Since one of the fire cause descriptions is ''Children'' and according to Robert Disbrow Jr. a career Firefighter/Investigator in New Jersey: "The child fire setter usually will set fires with a group and is influenced by peer pressure. Indicators are fires located near footpaths, trails, schools, or playgrounds". To address this we added a feature of the distance between each sample and the closest public school to that sample.

Distance to nearest Home Park:

Home parks are locations in which RVs and mobile homes can be parked, as such We have added the distance of each sample to closest.

Distance to nearest City and city name:

Since some types of fire are more likely to occur far from cities and some in cities (e.g: structure fire is more likely to occur in a city) we have added the distance to the closest city and since each city could have different fire causes distribution, we added the name of the closest city to sample.

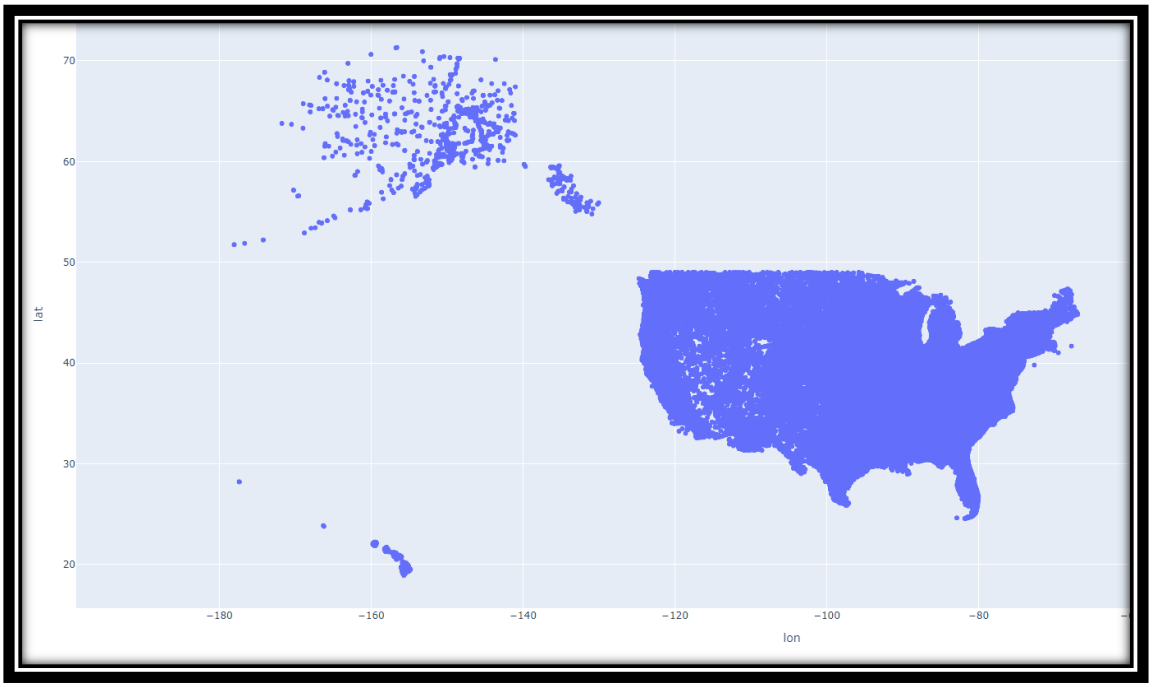
Weather features:

Since some types of fires are more likely to occur in different weather conditions (e.g: lightning fire cause is more likely to occur while a storm is raging) we have added the following features: daily min-temperature, max-temperature, avg-temperature, precipitation, and pan-evaporation.
To add those features we used the following method:

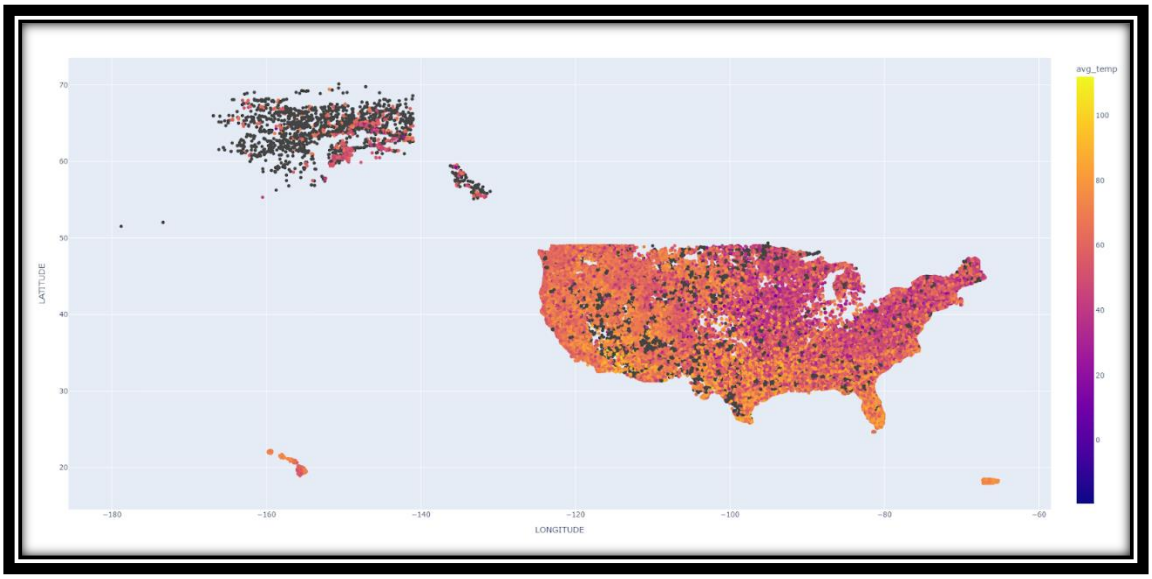
a. We created a database which gathers those features from many weather stations in the US for one day out of three days for each of the years 1992-2015.
b. For each sample we found the closest location in the database to the sample and added to our dataset the location's weather features from the closest date available in our database that was before the fire date discovery.

The reason we added weather information from before the fire discovery is to make sure there is no leakage from the fire itself to the weather information.

The graph below shows all the weather stations in the United States.



A literal Heatmap of the dataset using the average temperature feature in our dataset:



Distribution of smokers in the state in the year of the fire:

Since “Smoking” is a fire cause descriptor we added features representing the percent of adults in each state who never smoked, former smokers, smokes some days and smokes every day in the year of the fire.

Distribution of citizenship per state:
Two features representing the percent of citizens and non-citizens in the state compared to all inhabitants.

Race distribution per state:

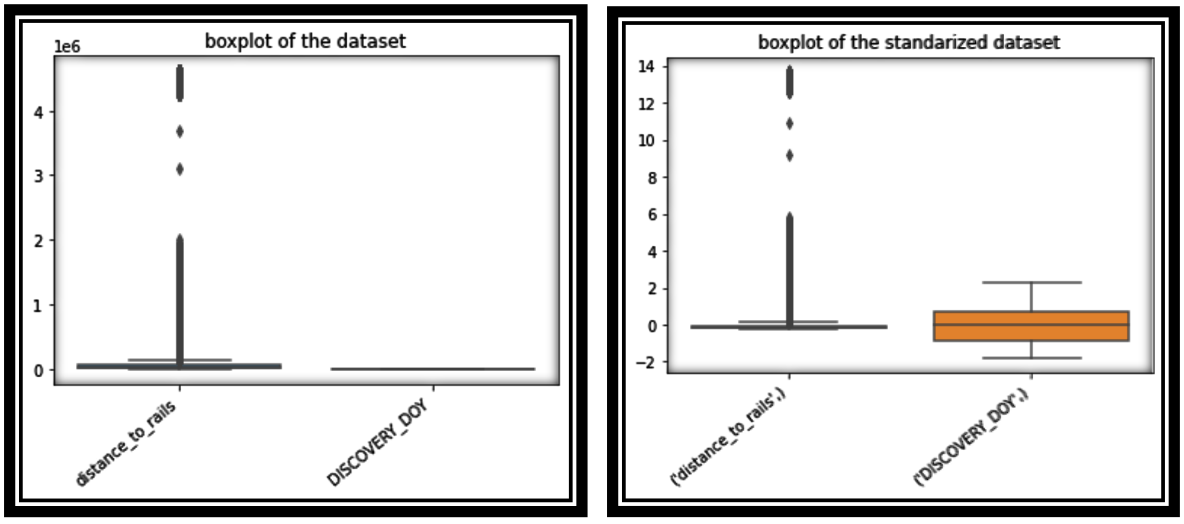
The percentage of white, black, hispanic, asian ect' in each state compared to all inhabitants.

State age demographics by year:
Features representing how many 0-18, 19-25, 26-34, 35-54, 55-64 and 65+ years old inhabitants each country has in the year of the fire.
We believe this feature might be important since one of the fire causes is "children" so states that have more children might have a higher chance of "children" fire cause.

Interactions:

Since from the age demographics by year we received the number of residents per country we used the interaction between the percentage of smokers and total residents to get the number of smokers for each state in the year of the fire.

Transformations:
While examining the data we saw that distance features have a lot more variance and a much bigger expectation then some of the other features, for example compared to the average temperature feature which could hurt model performance and feature importance analysis, as such we used sklearn's standard scaler to normalize the data.



For example the distance to rails feature in a box plot with discovery DOY before and after standardization using standard scaler.

Dimensionality reduction:

Using PCA we have tried to add features based on the PCA of all the features (excluding the labels) but after seeing that the PCA features resulted in decreased performance we have decided to not use them.

Categorical data:

To handle our categorical features: the state the fire was discovered in, the name of the closest city, land ownership code and fire size class we used one hot encoding since there is no order or scale between the values of those features.

After some testing we changed the fire size class into numbers each representing a single code in rising order since fire size classes have an order to them (e.g: grade "A" represents a smaller fire than grade "B" fire) for reasons we will go over in part 4.

Part 3: Tuning and Evaluating

Baselining:

In this section, note that because of time and resource considerations, we tested the following models with only 10% of the data. We know it's not ideal, but it's still good for the sake of comparing different models.

Prior to applying complex classification models to maximize prediction accuracy, baseline metrics were employed on pre-processed data as taught in class.
The first was to evaluate the performance of a prediction based on the mode of the training labels as follow:

```
In [4]: mode_prediction = np.ones_like(y_test)*y_train.mode()[0]
        print(classification_report(y_test,mode_prediction))
```

	precision	recall	f1-score	support
Arson	0.00	0.00	0.00	23344
Campfire	0.00	0.00	0.00	6387
Children	0.00	0.00	0.00	5148
Debris Burning	0.23	1.00	0.37	35842
Equipment Use	0.00	0.00	0.00	12324
Fireworks	0.00	0.00	0.00	931
Lightning	0.00	0.00	0.00	23246
Miscellaneous	0.00	0.00	0.00	26918
Missing/Undefined	0.00	0.00	0.00	13913
Powerline	0.00	0.00	0.00	1166
Railroad	0.00	0.00	0.00	2736
Smoking	0.00	0.00	0.00	4415
Structure	0.00	0.00	0.00	336
accuracy			0.23	156706
macro avg	0.02	0.08	0.03	156706
weighted avg	0.05	0.23	0.09	156706

This is of course not a good thing to do, and we got poor results on the test set.
The Random Forest classifier was selected as the baseline model and the results are as follows:

```
In [5]: rf= RandomForestClassifier(n_estimators=10)
        rf.fit(X_train,y_train)
        print(classification_report(y_test,rf.predict(X_test)))
```

	precision	recall	f1-score	support
Arson	0.48	0.54	0.51	23344
Campfire	0.41	0.31	0.35	6387
Children	0.26	0.15	0.19	5148
Debris Burning	0.51	0.64	0.57	35842
Equipment Use	0.34	0.27	0.30	12324
Fireworks	0.55	0.40	0.47	931
Lightning	0.73	0.80	0.77	23246
Miscellaneous	0.50	0.47	0.48	26918
Missing/Undefined	0.90	0.88	0.89	13913
Powerline	0.13	0.02	0.04	1166
Railroad	0.51	0.36	0.42	2736
Smoking	0.20	0.04	0.06	4415
Structure	0.24	0.04	0.06	336
accuracy			0.55	156706
macro avg	0.44	0.38	0.39	156706
weighted avg	0.53	0.55	0.54	156706

Based on the outcomes of the baseline, it is evident that a simple random forest model with limited number of estimators performs considerably well overall, particularly in accurately identifying instances of fire caused by lightning or for unknown reasons.

With the above observations, we can infer that the given classification problem is characterized by an imbalanced distribution of labels. Specifically, certain labels exhibit a significantly higher frequency than others, and there for a simple model's display on this data may be biased towards these labels.

There exist several techniques to address the problem of biased labeling, and we have attempted the following approaches. The first two techniques aim to increase the number of samples for labels that occur less frequently in the dataset.

1. Over-sampling

We used the Random Over Sampler approach, which duplicates samples for labels with low frequency, until all labels have the same frequency as the majority class (the class with most samples).
Using this method we have seen the macro average f1-score has risen as more instances of the minority classes have helped push the model into classifying instances as the minority class more frequently.
The f1-macro average rose from 0.41 to 0.44 using this method.

2. SMOTE algorithm

Additionally, we used the SMOTE algorithm, which generates synthetic samples for the minority class by interpolating between feature vectors of existing minority class samples. The algorithm creates new synthetic samples similar to the original minority class samples but not exact copies. Subsequently, it randomly selects these new samples and includes them in the training set, balancing the distribution of samples across all classes.
Using this method, while the macro-f1-average rose to the same value as when using random over sampler, we have observed a negligible decrease in the weighted f1-score.

3. Label Weights

Moreover, we experimented with the Label weights technique, which assigns relative importance to labels. This weight is used only when there is an overlap between a label and a feature. Ultimately, the final positioning of labels on the map is dependent on label and feature weights.
We didn't get any significant improvement using this method.

4. Under Sampling

We also considered the Under-sampling technique, which involves retaining all data in the minority class and reducing the size of the majority class. However, we chose not to use this technique as it could result in the removal of potentially important data without considering its usefulness to the analysis.

It is essential to note that there is a significant disadvantage to using these techniques, as the data may no longer accurately represent the real world. As a result, the analysis may be inaccurate and biased. Nevertheless, due to the complexity of our data and the challenge of correctly predicting its labels, we had to use some models and techniques that may introduce some bias.

Find the optimal model:

We proceeded with identifying the optimal model to commence work on based on its accuracy outcomes without parameter optimization.
We found out that the Random Forest classifier got the best results out of the 4 models we have tested:

AdaBoost Evaluation:

	precision	recall	f1-score	support
Arson	0.34	0.17	0.22	2229
Campfire	0.34	0.10	0.15	631
Children	0.17	0.02	0.04	544
Debris Burning	0.39	0.73	0.51	3577
Equipment Use	0.17	0.02	0.03	1237
Fireworks	0.17	0.31	0.22	89
Lightning	0.55	0.78	0.65	2258
Miscellaneous	0.39	0.39	0.39	2833
Missing/Undefined	0.53	0.41	0.46	1393
Powerline	0.15	0.04	0.06	112
Railroad	0.22	0.11	0.14	276
Smoking	0.00	0.00	0.00	451
Structure	0.06	0.05	0.05	41
accuracy			0.42	15671
macro avg	0.27	0.24	0.23	15671
weighted avg	0.37	0.42	0.37	15671

Random Forest Evaluation:

	precision	recall	f1-score	support
Arson	0.49	0.47	0.48	2229
Campfire	0.45	0.25	0.32	631
Children	0.32	0.08	0.12	544
Debris Burning	0.48	0.69	0.57	3577
Equipment Use	0.37	0.23	0.28	1237
Fireworks	0.55	0.33	0.41	89
Lightning	0.69	0.83	0.75	2258
Miscellaneous	0.50	0.49	0.50	2833
Missing/Undefined	0.89	0.87	0.88	1393
Powerline	0.00	0.00	0.00	112
Railroad	0.40	0.27	0.32	276
Smoking	0.21	0.01	0.02	451
Structure	0.00	0.00	0.00	41
accuracy			0.55	15671
macro avg	0.41	0.35	0.36	15671
weighted avg	0.52	0.55	0.52	15671

Nearest Neighbors Evaluation:

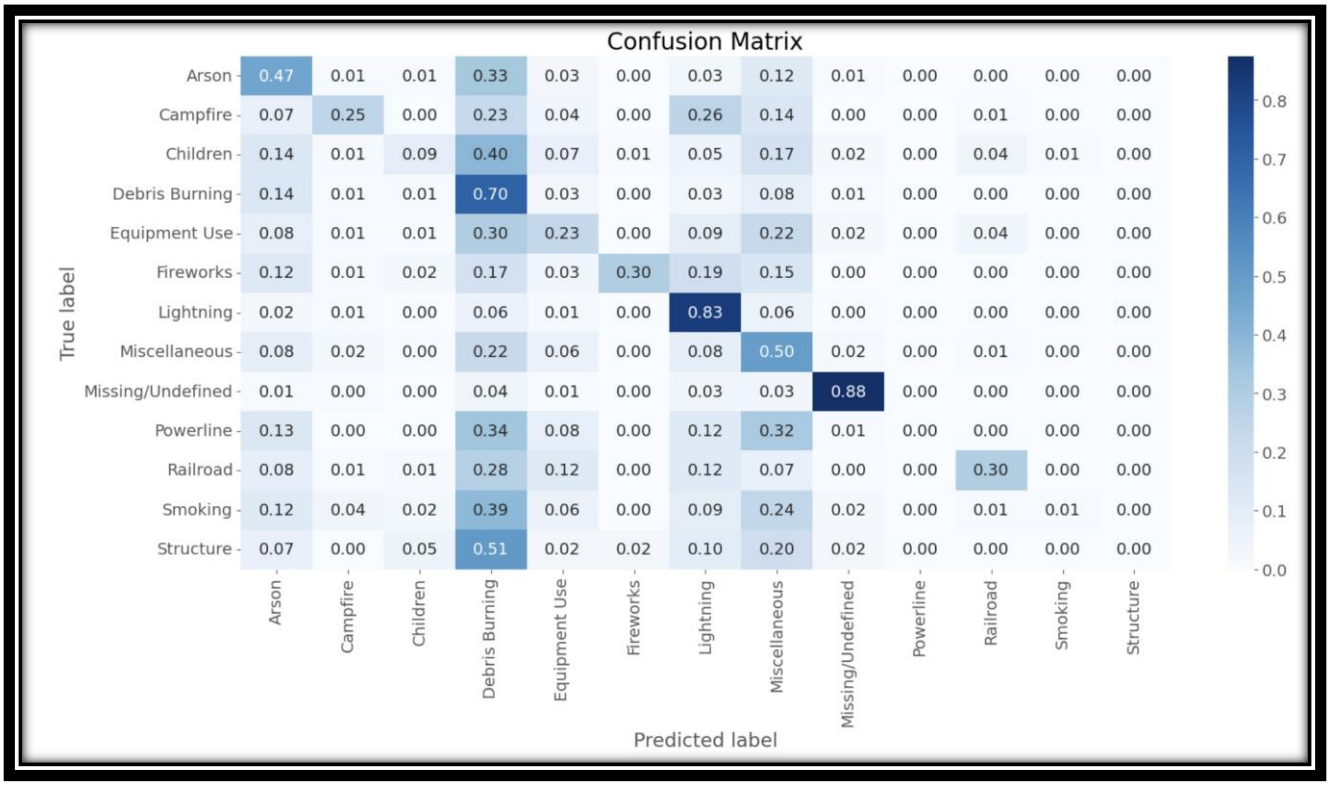
	precision	recall	f1-score	support
Arson	0.30	0.47	0.37	2229
Campfire	0.21	0.29	0.24	631
Children	0.11	0.12	0.12	544
Debris Burning	0.43	0.51	0.47	3577
Equipment Use	0.26	0.19	0.22	1237
Fireworks	0.29	0.22	0.25	89
Lightning	0.69	0.67	0.68	2258
Miscellaneous	0.48	0.32	0.38	2833
Missing/Undefined	0.87	0.80	0.83	1393
Powerline	0.17	0.04	0.06	112
Railroad	0.37	0.17	0.23	276
Smoking	0.10	0.02	0.03	451
Structure	0.00	0.00	0.00	41
accuracy			0.44	15671
macro avg	0.33	0.29	0.30	15671
weighted avg	0.45	0.44	0.44	15671

Neural Net Evaluation:

	precision	recall	f1-score	support
Arson	0.47	0.37	0.41	2229
Campfire	0.43	0.21	0.28	631
Children	0.18	0.03	0.04	544
Debris Burning	0.45	0.75	0.56	3577
Equipment Use	0.34	0.08	0.13	1237
Fireworks	0.54	0.21	0.31	89
Lightning	0.66	0.80	0.72	2258
Miscellaneous	0.49	0.46	0.47	2833
Missing/Undefined	0.76	0.87	0.81	1393
Powerline	0.00	0.00	0.00	112
Railroad	0.38	0.34	0.36	276
Smoking	0.00	0.00	0.00	451
Structure	0.00	0.00	0.00	41
accuracy			0.52	15671
macro avg	0.36	0.32	0.32	15671
weighted avg	0.48	0.52	0.48	15671

Confusion Matrix:

Now that we have a classifier to work with, and trained model based on it, we would like to evaluate its performance. First, let's visualize with a confusion matrix:



From the confusion matrix above, we conclude that the random forest model tends to classify fires caused by structure, children and smoking as debris burning and in general doing badly in predicting correctly fires caused by children, power line, smoking and structure.

On the next part we'll address these confusions.

Hyper-Parameter Optimization: Random Search

We now have the basic model which we want to improve and know its weaknesses regarding too confusing between the labels.
The next thing we wished to do was hyper-parameter optimization:

To find the best hyper parameters, we first tried using random search as we have seen in class:
After conducting a grid on a random forest classifier, we used sklearn's RandomizedSearchCV on the grid and got the following results:

Best Params: {'random_state': 666, 'n_estimators': 1800, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 55, 'bootstrap': True}

We compared its performance to that of a standard random forest model.
Surprisingly, our results indicate that sometimes simpler models can outperform more complex ones, as the random forest simple model performed better comparing to the grid search.
Note that although we used only 10% of our data for training and testing our models, the computational resources required by the random search classifier were too excessive, and we were unable to obtain its results.

Trying some more complex models:
We have developed more complex optimization functions, which involve cross validation and investigate the potential benefits of using dimensionality reduction. Specifically, we have created four separate resumable functions, each designed to evaluate a different classification model that have selected based on their perceived importance. All four functions operate in the same manner, as outlined below:
The function *'specify for each model'* performs a grid search over hyperparameters and returns the best combination of these parameters for a given the dataset and target labels.
The function first initializes the lists of values of the hyperparameters. It then sets up a loop to iterate over all combinations of these hyperparameters.

We have tried to use K-Fold cross-validation to split the data into training and testing sets to provide a more robust estimate of the model's performance than a single train-test split. By using multiple train-test splits, K-Fold cross-validation allows us to better estimate the variance of the model's performance and reduce the risk of overfitting).
Unfortunately, due to lack of computing power and memory imitations, we could not run this model using cross validation split on all of our data.

Then the function fits the classifier on the training data with the given hyperparameters. If dimensionality reduction is used (given n_components is not None), PCA is applied to reduce the dimensionality of the training and testing data. The function then computes the accuracy of the classifier on the testing data and records it in a list. After both folds are processed, the function takes the mean of the accuracy over the folds as the performance metric for this combination of hyperparameters.

Unfortunately, those functions performed no better than the regular search grid and random grid even though they seem to be more sophisticated.

Next try - Confusion Matrix Based Mixture of Experts

To improve the prediction performance, we propose using a Confusion Matrix Based Mixture of Experts approach, in which we cluster similar classes based on their confusion scores as determined by the confusion matrix. This allows us to identify which subsets of classes the initial model has difficulty distinguishing between.
We create subsets of the original dataset for each cluster, containing only examples from the classes in the cluster. We then perform model selection for each of these subsets, in order to choose a specialized model for each cluster. At test time, we first use the initial model to make an initial prediction for an example. Based on this prediction, we choose a specialized model from the subset of classes to make a final prediction for the example.
By selecting specialized models for subsets of classes with high confusion scores, we can improve the model's overall performance. This approach allows us to tailor the model to each cluster of similar classes, rather than treating all classes equally, which can result in suboptimal performance.
We have constructed a method that initializes the parameters of the model (finally we chose random forest with default parameters but tested the program with other parameters as well, which did not perform as good as the random forest).

Note that the Confusion Matrix Based Mixture of Experts model we have constructed performed a bit better than the original random forest model, but not significantly better. We think that the reason for this might be as follows:
As the random forest model with no depth limitation searches for the best cluster (or, originally, label) until it gets to the best unique result for each sample. Since on our dataset, random forest by itself already reaches f1 weighted average of close to 1, the increased model robustness achieved by the confusion matrix-based mixture of experts did not improve the performance of the model as much as we assumed it will.

The results of this model combined with random over sampling gave us the best results so far:

```
=== starting with clustering threshold: 0.9===
{0: [0, 1, 4, 5, 6, 7, 11], 6: [2], 5: [3], 2: [8], 4: [9], 3: [10], 1: [12]}
found 7 cluster
training random forest...
training logistic regression...
fitting expert for cluster 0
                  precision    recall  f1-score   support

           Arson       0.56      0.55      0.56     23367
        Campfire       0.48      0.42      0.45      6333
        Children       0.32      0.18      0.23      5139
   Debris Burning       0.55      0.66      0.60     35804
    Equipment Use       0.37      0.35      0.36     12379
        Fireworks       0.54      0.54      0.54       968
        Lightning       0.75      0.84      0.80     23237
    Miscellaneous       0.55      0.50      0.52     27028
Missing/Undefined       0.87      0.91      0.89     13876
        Powerline       0.25      0.09      0.14      1190
         Railroad       0.50      0.49      0.49      2751
          Smoking       0.22      0.05      0.08      4325
        Structure       0.20      0.05      0.07       309

         accuracy                           0.59    156706
        macro avg       0.47      0.43      0.44    156706
     weighted avg       0.57      0.59      0.58    156706
```

Part 4: Feature Importance and Outlier Detection

As we showed and discussed earlier in the project - we have added many more features to the original dataset. We chose those added features by intuition: We've added the features that we thought would help us most in completing our task, which is predicting the cause of the fire. Nothing scientific up to this point.

In class we've learned a few methods to determine what features are more important than others, and that's what we'll do here.

Since the chosen model is a Random Forest Classifier, we will use feature importance methods for tree-based models.

Before we make any changes to the data, just so we would be able to compare the results, the f1-macro average using those features and our trained model was 0.40, and the f1-accuracy 0.57:

	precision	recall	f1-score	support
Arson	0.55	0.49	0.52	23306
Campfire	0.54	0.28	0.37	6505
Children	0.41	0.11	0.17	5114
Debris Burning	0.50	0.74	0.59	35694
Equipment Use	0.44	0.23	0.30	12401
Fireworks	0.68	0.41	0.51	982
Lightning	0.73	0.85	0.78	23266
Miscellaneous	0.49	0.52	0.51	26957
Missing/Undefined	0.88	0.84	0.86	13797
Powerline	0.42	0.02	0.04	1211
Railroad	0.54	0.41	0.47	2755
Smoking	0.33	0.02	0.03	4407
Structure	0.57	0.04	0.08	311
accuracy			0.57	156706
macro avg	0.54	0.38	0.40	156706
weighted avg	0.56	0.57	0.55	156706

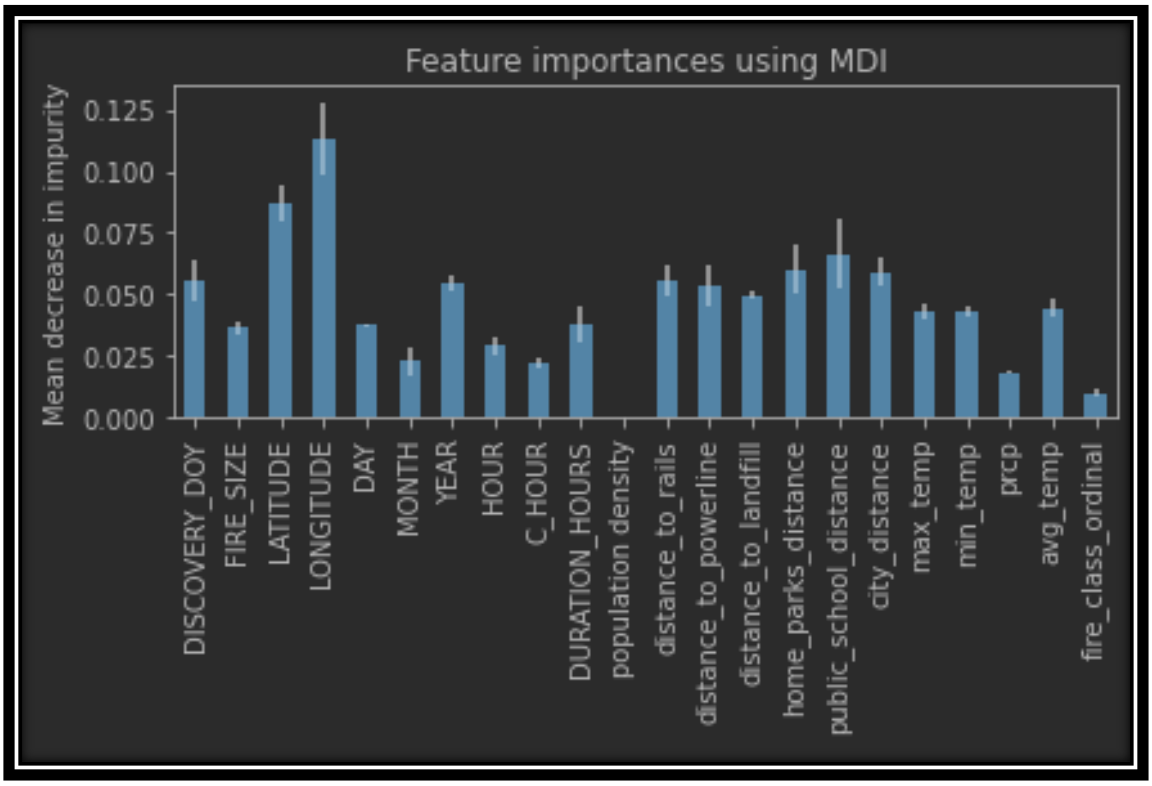
<u>Mean Decrease in Impurity</u>

First problem we ran into was the following: There is a problem to do feature importances while using 'one hot encoding' for features. That is simply because when converting values to one-hot-encoding, we are getting 'new features' that represent one instance of the class. So, one option was to reverse the one hot encoding - but that is not elegant at all and can't be done on a systematic way (in ex3 we did it, but it was possible since the data was not too big and we only worked with 5 categories).

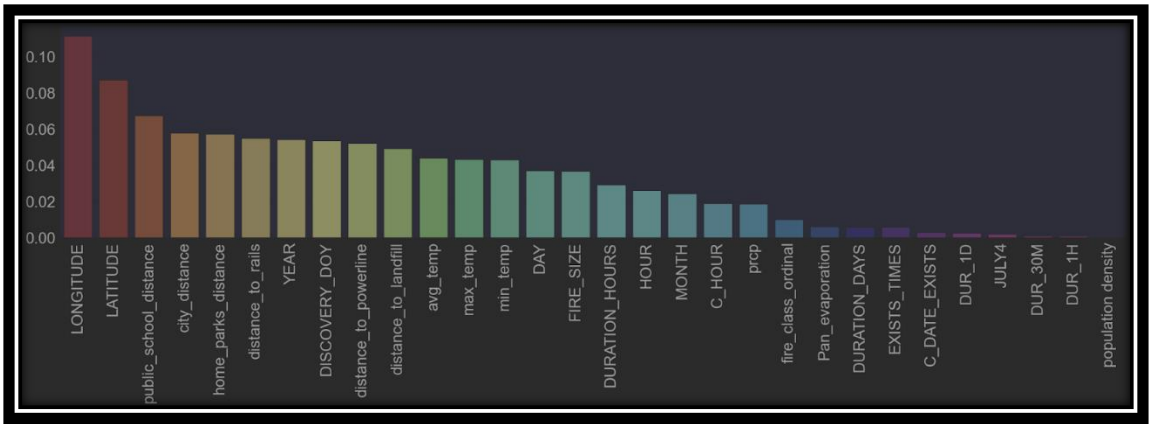
The other solution was, first of all, try not to use one hot encoding where we can, and second - to run the feature importance methods on the features that are not

one-hot encoded. For example, we realized that 'FIRE_CLASS_SIZE' can be an ordinary feature instead of one-hot, so we converted it, and so on.

So, with that in mind - we got the following results:



We've used sns barplot to show them in a sorted way:



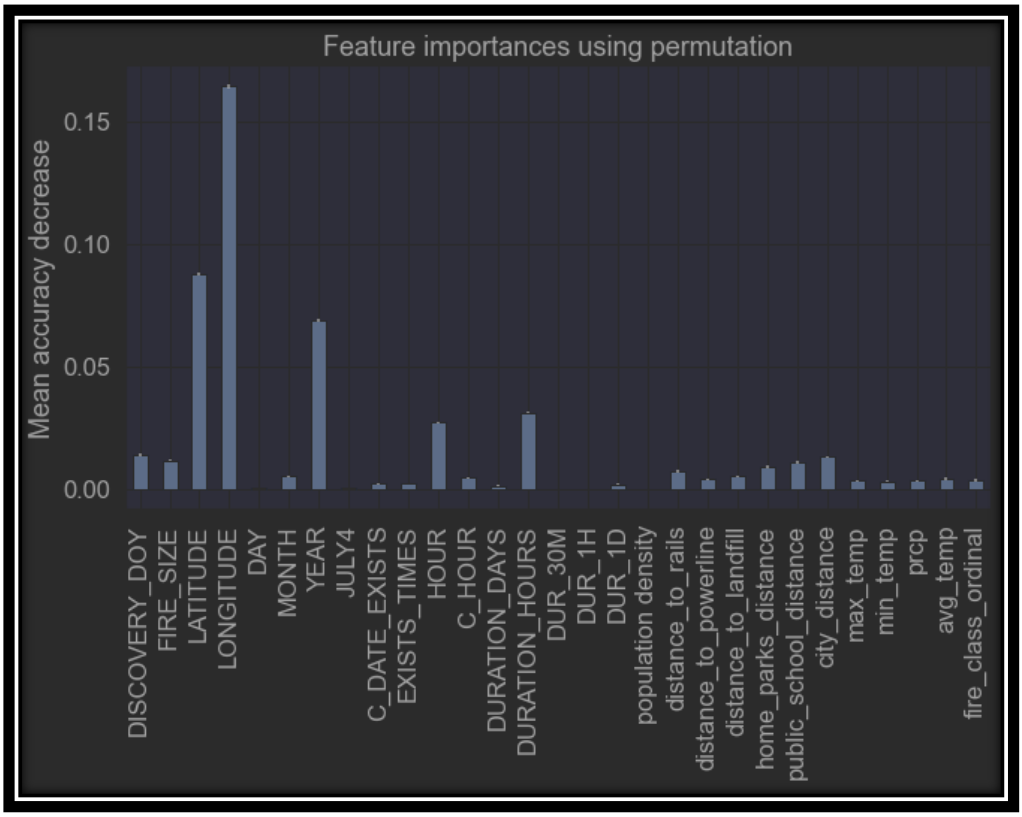
Now, we deleted the non-important features that is: ['DURATION_DAYS', 'JULY4', 'C_DATE_EXISTS', 'EXISTS_TIMES','DUR_30M', 'DUR_1H', 'DUR_1D','population density'], and checked if we did better, and turns out that we did. Not by a huge amount, but the recall seems to improve, and so the f1-accuracy went up to 0.58:

	precision	recall	f1-score	support
Arson	0.55	0.50	0.52	23301
Campfire	0.52	0.27	0.36	6406
Children	0.39	0.11	0.17	5076
Debris Burning	0.50	0.74	0.60	35868
Equipment Use	0.42	0.23	0.29	12173
Fireworks	0.67	0.39	0.49	1018
Lightning	0.73	0.85	0.78	23108
Miscellaneous	0.50	0.52	0.51	27174
Missing/Undefined	0.89	0.84	0.86	14043
Powerline	0.49	0.02	0.04	1179
Railroad	0.55	0.41	0.47	2768
Smoking	0.24	0.01	0.02	4276
Structure	0.59	0.03	0.06	316
accuracy			0.58	156706
macro avg	0.54	0.38	0.40	156706
weighted avg	0.56	0.58	0.55	156706

We have read that impurity based feature importances can be misleading for high cardinality features, which can be the case in this dataset, so it's good to try another method as well.

Feature Permutation

This is the second method that we have learned for tree-based feature importances. Using it we got:



We can see that we got some changes here: year became much more important than other features that in the first method were similar in importance. However, it looks like the least important features are agreed more-or-less by both methods. This time we did not get any improvement:

	precision	recall	f1-score	support
Arson	0.55	0.49	0.52	23299
Campfire	0.52	0.29	0.37	6365
Children	0.42	0.12	0.18	5154
Debris Burning	0.50	0.74	0.60	35802
Equipment Use	0.43	0.23	0.30	12388
Fireworks	0.63	0.39	0.48	968
Lightning	0.72	0.85	0.78	23248
Miscellaneous	0.49	0.51	0.50	27101
Missing/Undefined	0.88	0.84	0.86	13712
Powerline	0.30	0.01	0.03	1223
Railroad	0.53	0.41	0.46	2768
Smoking	0.32	0.01	0.03	4373
Structure	0.55	0.04	0.07	305
accuracy			0.57	156706
macro avg	0.53	0.38	0.40	156706
weighted avg	0.56	0.57	0.55	156706

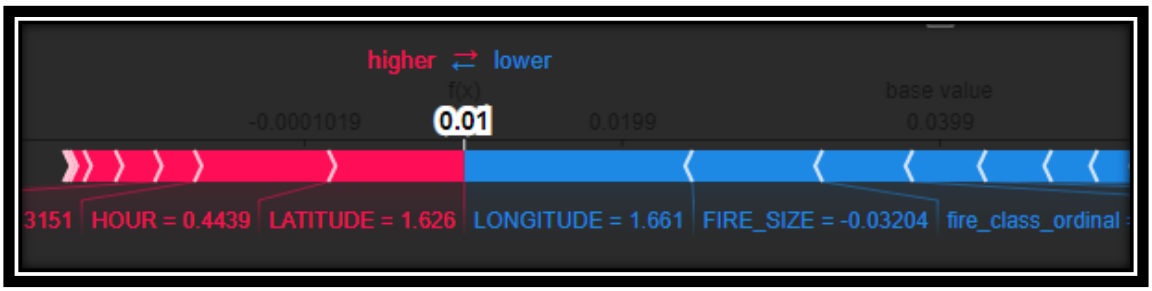
So, we decided to remove the non-important features that are agreed by both models only.

SHAP Values

We'll try to find out what the importance of each feature is for our model.
With SHAP we are doing it by checking how much a certain value of a certain feature will contribute to the model. Here we can see the results of 2 different samples:

Sample1:

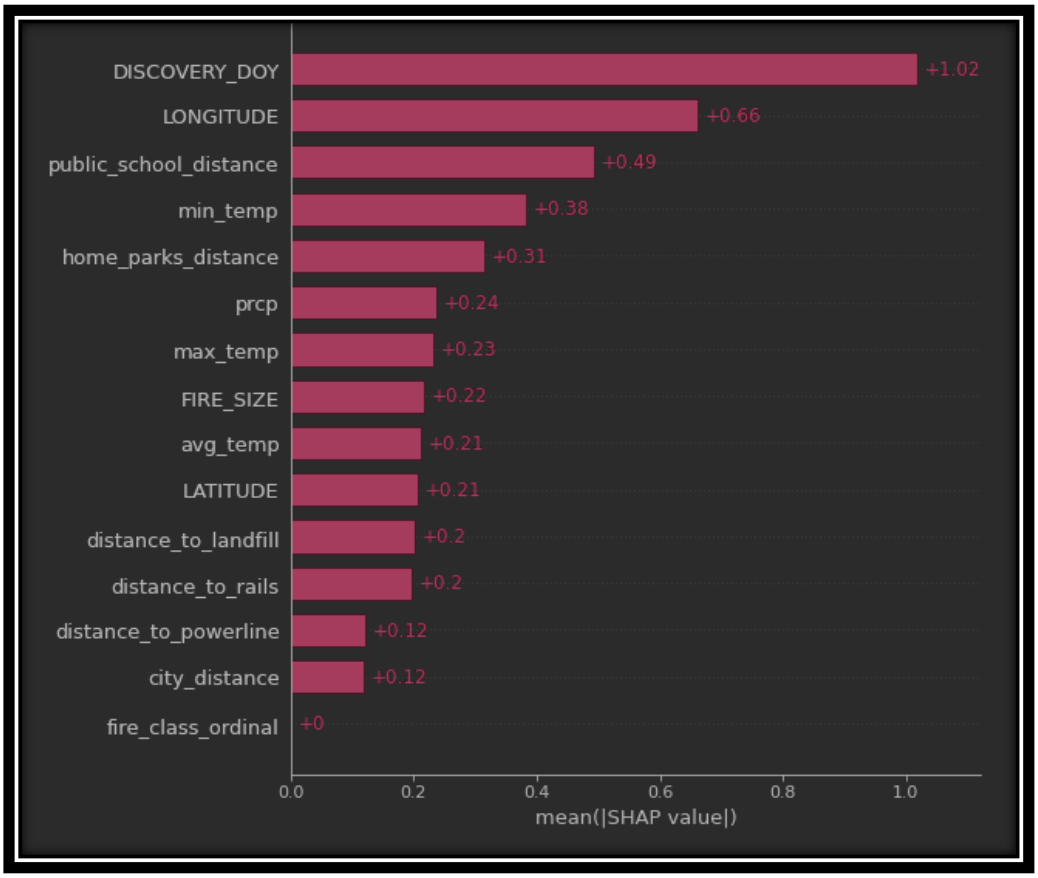
Sample 2:



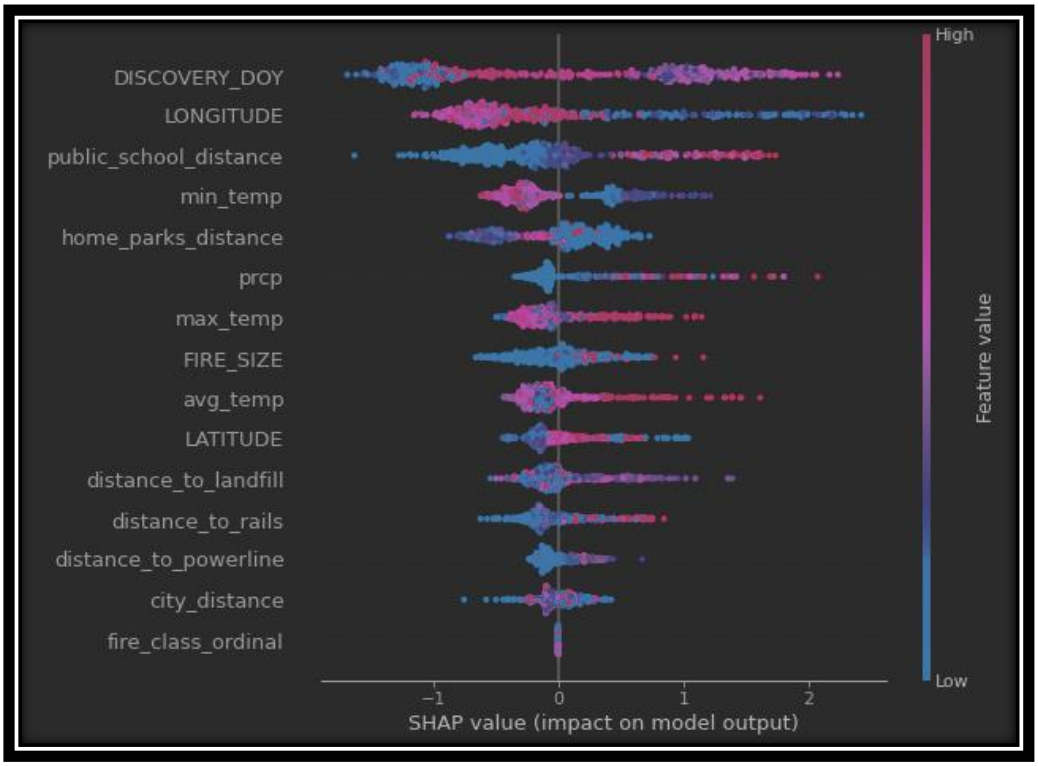
As expected, the SHAP value can change within different samples, but we can see that overall, the important features that we got with the 'Mean Decrease In Impurity' method are the ones that dominate in these 2 samples as well (Longitude, Latitude, Fire Size…).

To see the whole picture, and not only for 2 samples, we used xgboost as seen in class. We can see in the 2 graphs below, only the 20 most important features:

Graph where SHAP is in in absolute value:

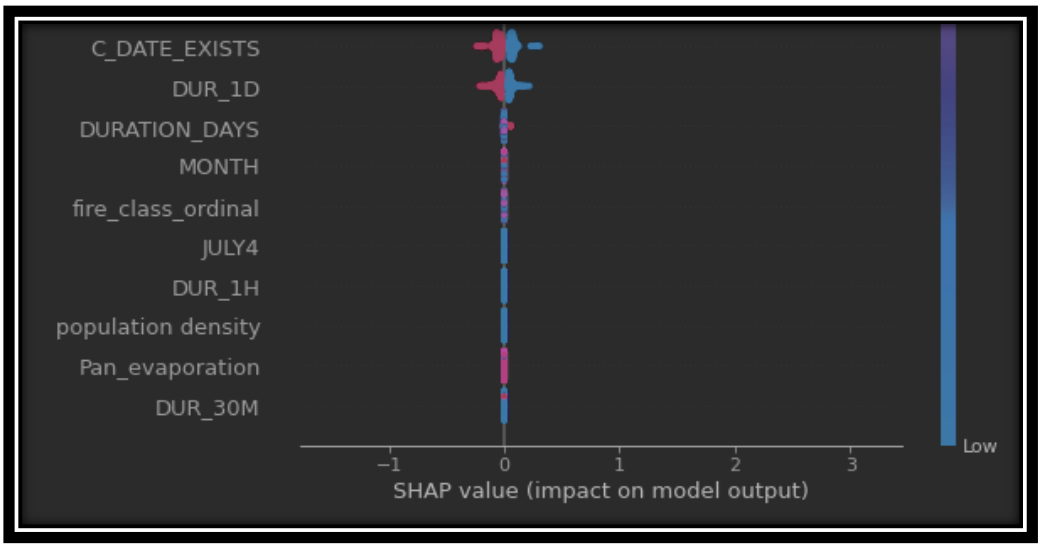


Graph with SHAP value:



So, as we can see, there is no explicit agreement on the most important features between the SHAP method and the previous ones, since here we can see features that was not as important as in 'Mean Decrease in Impurity' method. However, the important features from before are still quite important here as well (Longitude, Latitude, Fire Size…).

On the other hand, it seems that there is an agreement over what features are not important! In the next figure we can see the least important features as far as SHAP tells us:



Those are the same features that 'Mean Decrease In Impurity' and 'Feature Permutation' considered as not important.

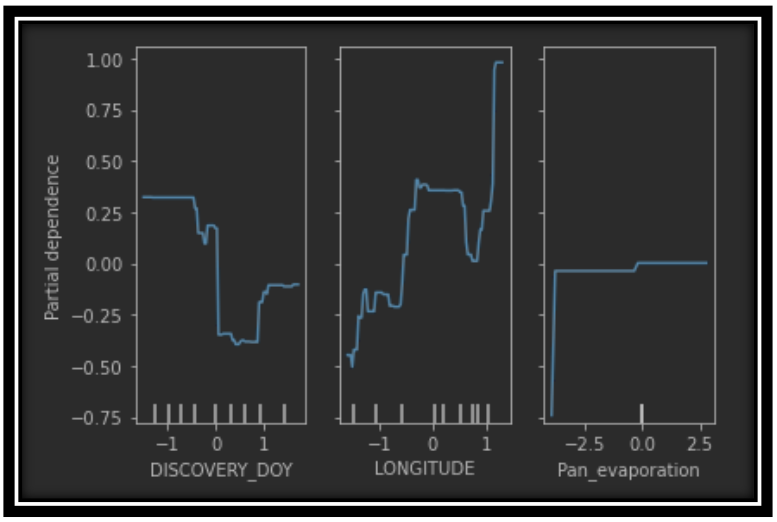
So, while not (or less) helping us with the important features, we are now more confident in what features we can drop.

<u>A few notes regarding important features:</u>
A lot of conclusions we've made here should be carefully considered. That's because the term 'importance' is quite ambiguous. We'll give an example to clarify our point: We have the 'july_4' feature, which is a binary feature that tells us whether the fire happened in USA's Independence Day or not. On this day, any fire that occurred has a high chance of coming from fireworks. So, while this feature is not broadly important for the model, as seen in the graphs above (since most of the fires in the database didn't occur in that day), it is very important for predicting the cause on this day, and also with very high probability to make a correct prediction. In a way, this feature is sort of a 'free answer' to the question "did this fire originated from fireworks" – so it's not at all a non-important feature.
To prove this point, we found that 40% of fires caused by fireworks happened on 4th of July, while less than 1% of fires caused by debris (which is the most common cause in the data) happened on that day.

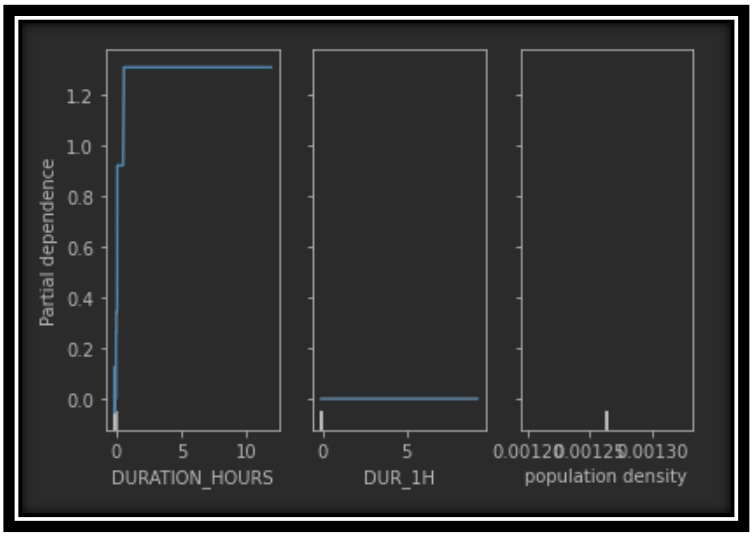
Partial Dependency

In order be more confident in the importances of the features we investigated, we did partial dependency for 3 features:
A feature that considered very important in the SHAP method - DISCOVERY_DOY,
A feature that considered very important in the 'impurity' and 'permutation' methods - LONGITUDE,
And a feature that was not important to all methods - 'pan_evaporation'.

Here are the results:



The above graph confirms once again what we know by now - DISCOVERY_DOY and LONGITUDE features have effect on the predictions (while in this case LONGITUDE has more effect), and 'pan_evaporation' has almost no effect at all.The results are the same for other agreed upon non-important features, for example:



Outlier Detection

We tried to find outliers using isolation forest, as we did in ex3.
We ran the isolation forest on our data (after preprocessing), and took out all the samples that the isolation forest considered as outliers (2440 in total, out of 468116 samples in the train set, which is not much at all).

Then, we trained the model again. Below we can see the results with and without the outliers:

With Outliers:

	precision	recall	f1-score	support
Arson	0.57	0.53	0.55	23417
Campfire	0.53	0.34	0.42	6309
Children	0.41	0.12	0.19	5131
Debris Burning	0.52	0.73	0.61	35707
Equipment Use	0.41	0.27	0.33	12314
Fireworks	0.64	0.43	0.51	933
Lightning	0.75	0.85	0.80	23209
Miscellaneous	0.53	0.53	0.53	27185
Missing/Undefined	0.90	0.90	0.90	13844
Powerline	0.34	0.04	0.07	1146
Railroad	0.54	0.44	0.49	2803
Smoking	0.30	0.03	0.05	4374
Structure	0.52	0.04	0.08	334
accuracy			0.60	156706
macro avg	0.53	0.41	0.42	156706
weighted avg	0.58	0.60	0.57	156706

Without Outliers:

	precision	recall	f1-score	support
Arson	0.57	0.53	0.55	23417
Campfire	0.53	0.34	0.42	6309
Children	0.41	0.13	0.19	5131
Debris Burning	0.52	0.73	0.61	35707
Equipment Use	0.42	0.27	0.33	12314
Fireworks	0.63	0.43	0.51	933
Lightning	0.75	0.86	0.80	23209
Miscellaneous	0.53	0.53	0.53	27185
Missing/Undefined	0.90	0.91	0.90	13844
Powerline	0.34	0.04	0.07	1146
Railroad	0.55	0.45	0.49	2803
Smoking	0.32	0.03	0.05	4374
Structure	0.48	0.03	0.06	334
accuracy			0.60	156706
macro avg	0.53	0.41	0.42	156706
weighted avg	0.58	0.60	0.57	156706

While there are minor differences in the context of specific features, there is no improvement in the weighted f1 score.

Conclusions and where to go from here:

In fire cause investigation generally the most important factor in deciding the fire cause is the forensic examination of the origin point of the fire. By combing through where the fire started from investigators could discover numerous things:
How many sources the fire originated from? Are there any leftovers in the area from a campfire? cigarette butts or firework residue? Was there a fire accelerant involved in the fire ignition? etc.
As such, one of the biggest challenges we encountered in this project is the lack of such forensic information. This limitation resulted in it being very hard to create features that will differentiate between different fire causes.
To create better and stronger ML methods for predicting fire causes we believe collecting such forensic data and incorporating it into the WildFire dataset could improve model performance.

The second challenge we faced is hardware limitations. Whether it was the limited number of cores to accelerate multiprocessing or limited RAM, there were some cases were testing some models or trying to add new features required us to use suboptimal methods to reduce runtime or at times even had to give up entirely on it. As such, using computers with stronger hardware and creating larger databases for additional features would've allowed for more in-depth feature and model exploration.

Additionally, to get better performance some exploration of Neural Networks models could result in better models since NN models are the best performing models in today's cutting-edge high complexity ML problems. But designing NNs is a hard task that requires both high computational power and profound understanding of NNs.

In conclusion, during this project we got to experience the hardships and challenges of real life data science problem solving. While this project could be improved upon, we are happy with our final results and look forward to seeing the improvements the following years will bring to the field.