Applied Competitive Lab in Data Science - Final Report

Peleg Reichman
Adam Shtrasner
Asif Kagan

Data Exploration

Irrelevant Features

After investigating the meaning of each feature. We decided to drop the following features:

- **FIRE_CODE, ICS_209_INCIDENT_NUMBER, ICS_209_NAME, MTBS_ID, MTBS_FIRE_NAME, COMPLEX_NAME** - those features don't seem to be informative, and in addition they have a large amount of null values - with **FIRE_CODE** having 518k null values, and more than 90% of the values in the rest of the features are null.

- **SHAPE** - contains values that are not useful.

- **FIPS_NAME, NWCG_REPORTING_UNIT_NAME -** containing above 1500 unique values, which means these features complicate the model's complexities greatly, and risks overfitting, so we decided to drop them.

- **FIPS_CODE, SOURCE_SYSTEM_TYPE, NWCG_REPORTING_NAME, SOURCE_REPORTING_UNIT, SOURCE_REPORTING_UNIT_NAME, FIRE_NAME** - Contains information that does not seem to be helpful and relevant for predicting cause of fires.

- **COUNTY, STATE** - as we are already using the **LONGITUDE** and **LATITUDE** features, we don't need the information about the county and the state.

- **OBJECTID, FOD_ID, FPA_ID, NWCG_REPORTING_UNIT_ID, LOCAL_FIRE_REPORT_ID, LOCAL_INCIDENT_ID** - those are identifier values, and therefore not informative.

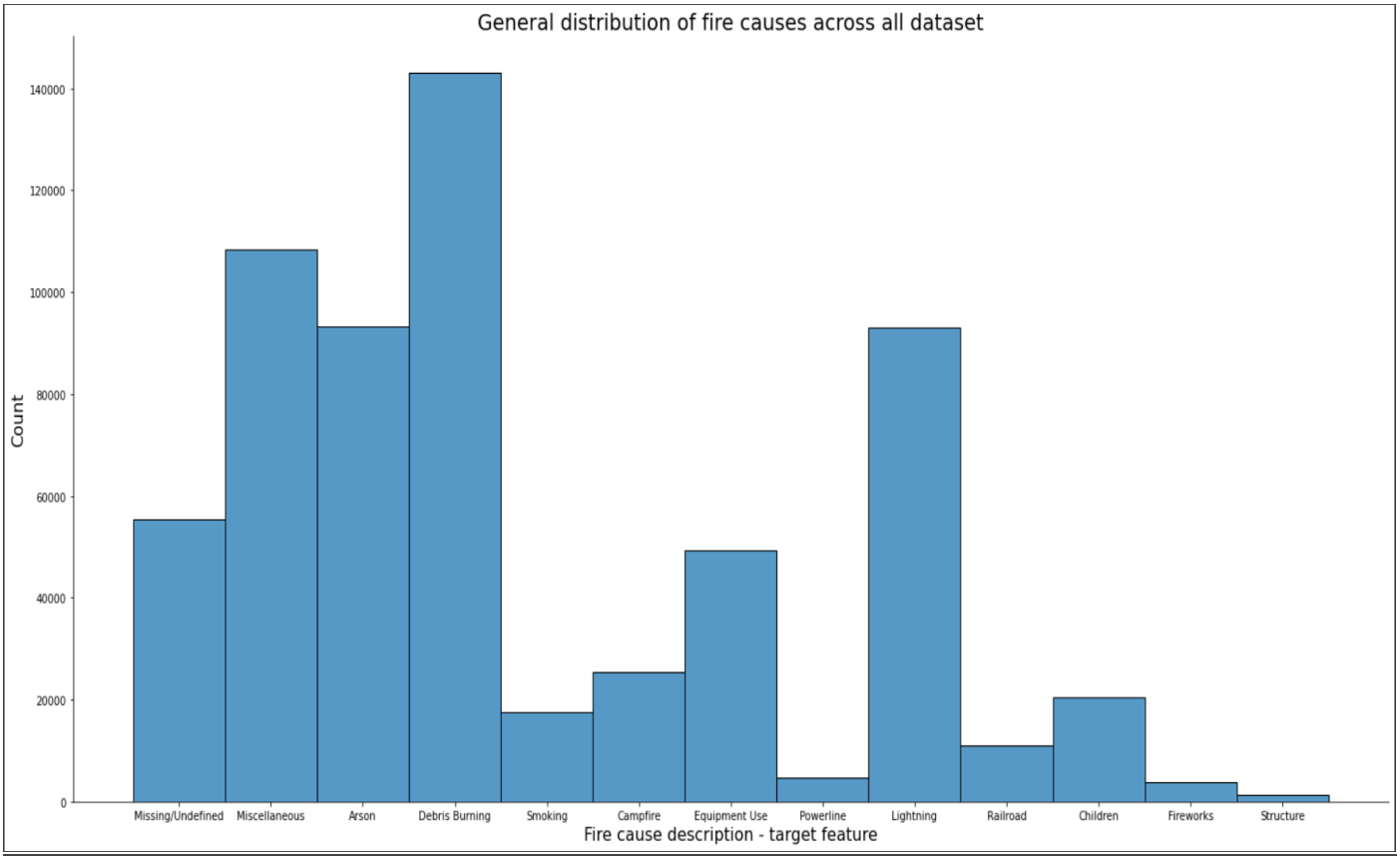
Redundant Features

We dropped features which can be comprised of other features (that we decided to use) and therefore redundant:

- **OWNER_CODE** - directly derived from **OWNER_DESCR.**
- **FIRE_SIZE** - directly derived from **FIRE_SIZE_CLASS.**
- **STAT_CAUSE_CODE** - directly derived from **STAT_CAUSE_DESCR**.

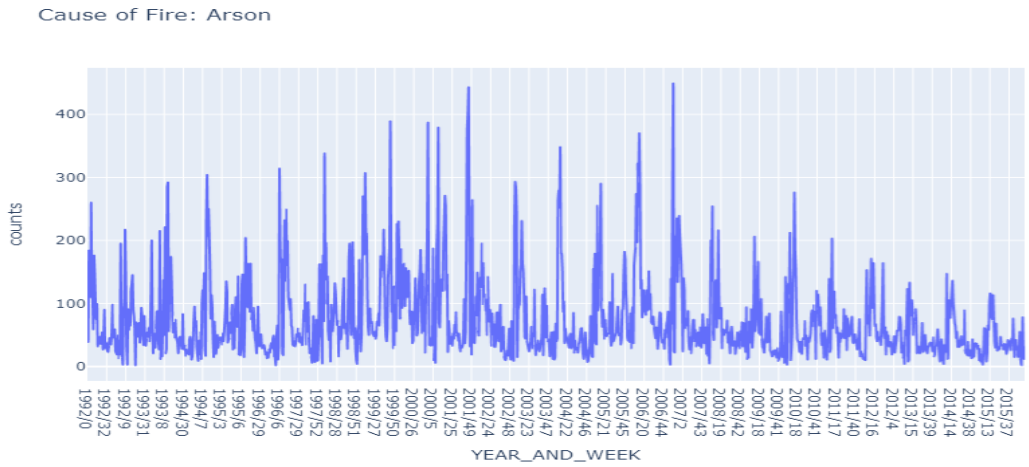
Visualization

General distribution of fire causes across all dataset

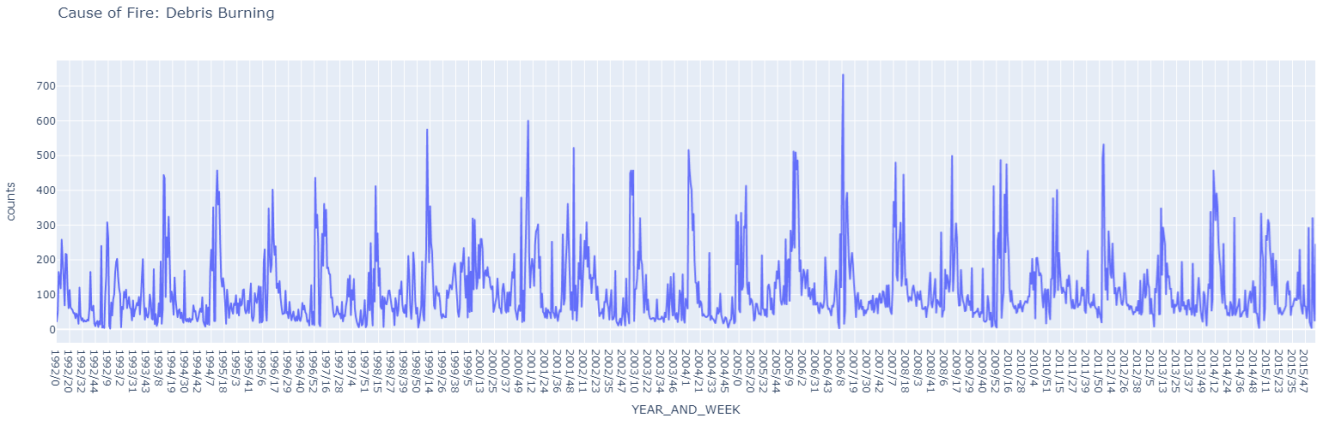


As seen in the figure above the target feature is highly imbalanced throughout the dataset.
This insight guided us while building the pipeline, from feature engineering to model selection, as it was our main challenge in this project.

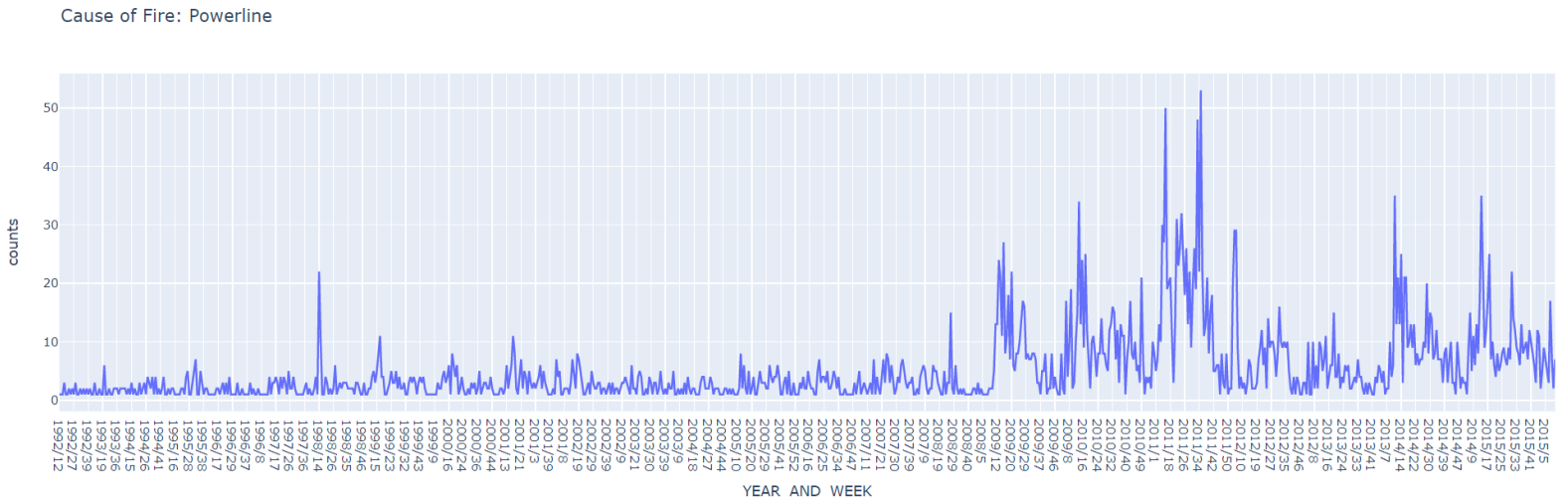
Trends in Fire Incidents Per Week of the Year According to Fire Cause



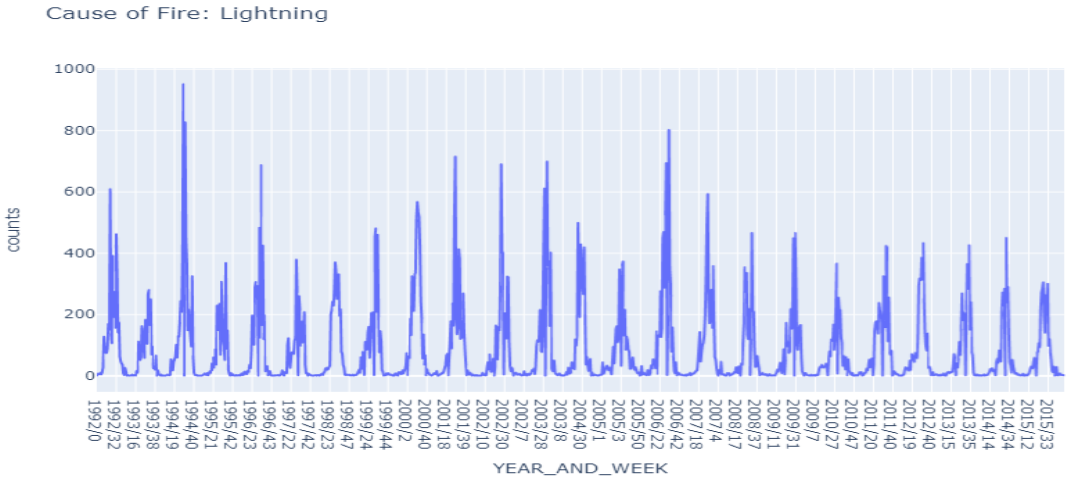
- Fire incidents caused by arson decreased in average after 2014.



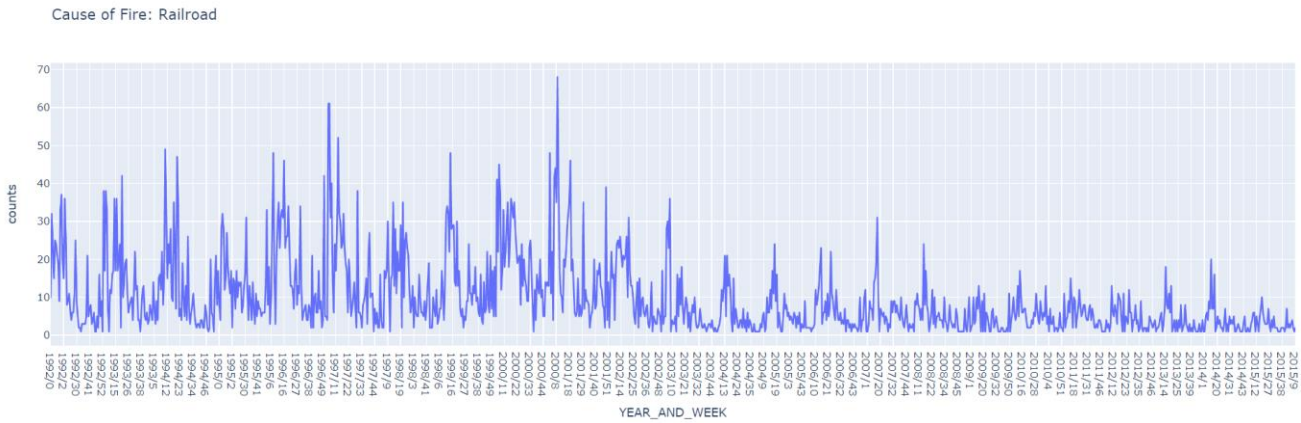
- Fire incidents caused by Debris Burning occur mainly after the winter, this trend is logical, the weather in those weeks (April and May), when homeowners are cleaning up from the winter months, and when the majority of vegetation is void of any appreciable moisture making them highly flammable.



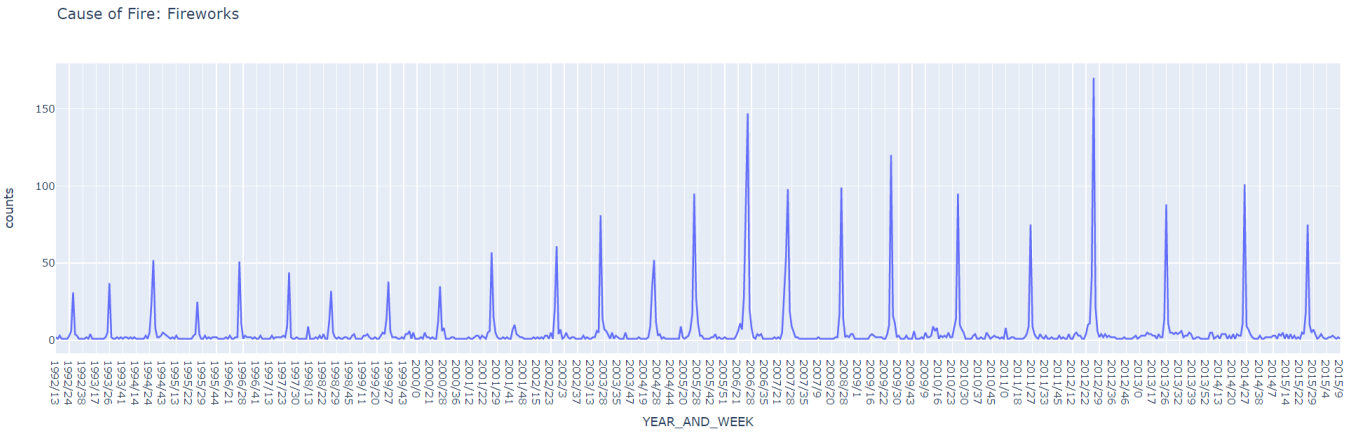
- Fire incidents caused by Powerline are on average from 2009, after searching a bit online, the reason is aging of old power lines, as a high percentage of US power lines were deployed around the early 90's.



- Fire incidents caused by lightning, mostly don't occur during the end of each year and in the first 4 months of the consecutive year, the reason is weather and season conditions which effects frequency of lightning storms and also vegetation which is green and wet in those months.

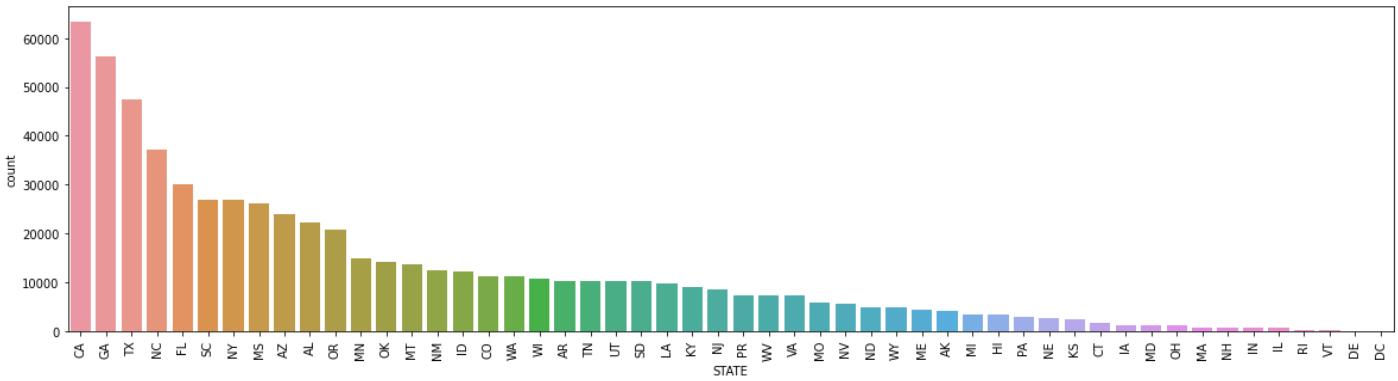


- As opposed to the trend in fires caused by power lines, here we see a decrease in fires from around 2006 when it comes to fires caused by railroads, due to technology improvement.



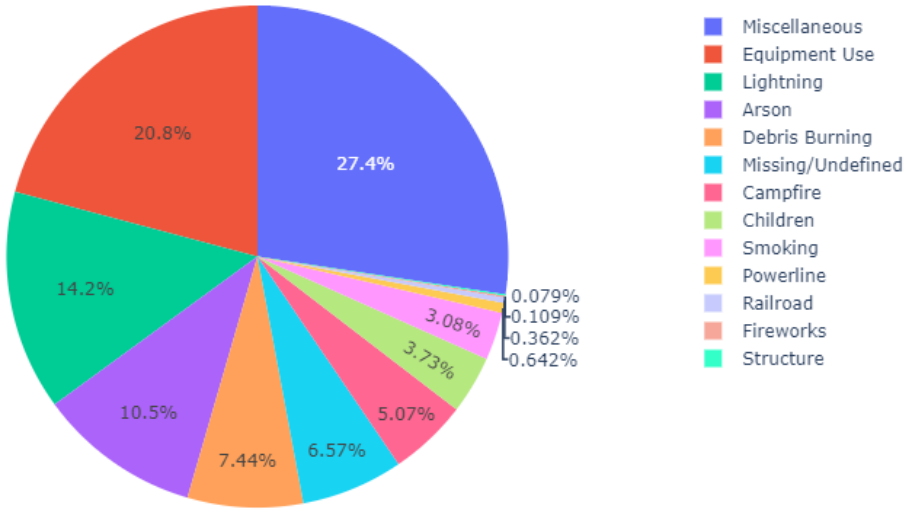
- We can see that in weeks at the start of July, there is a sharp increase in fire incidents caused by fireworks, which makes sense since there are alot of fireworks during the 4th of July.

Distribution of Fire Incidents Per State

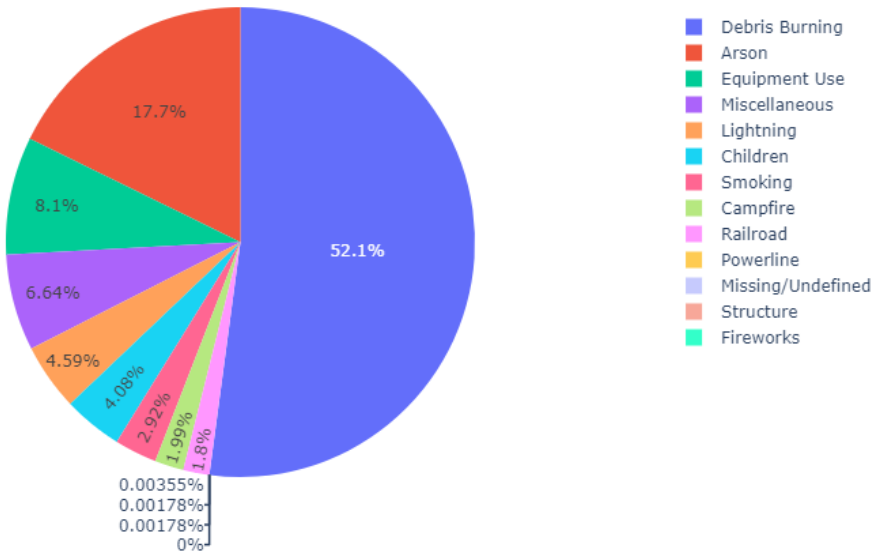


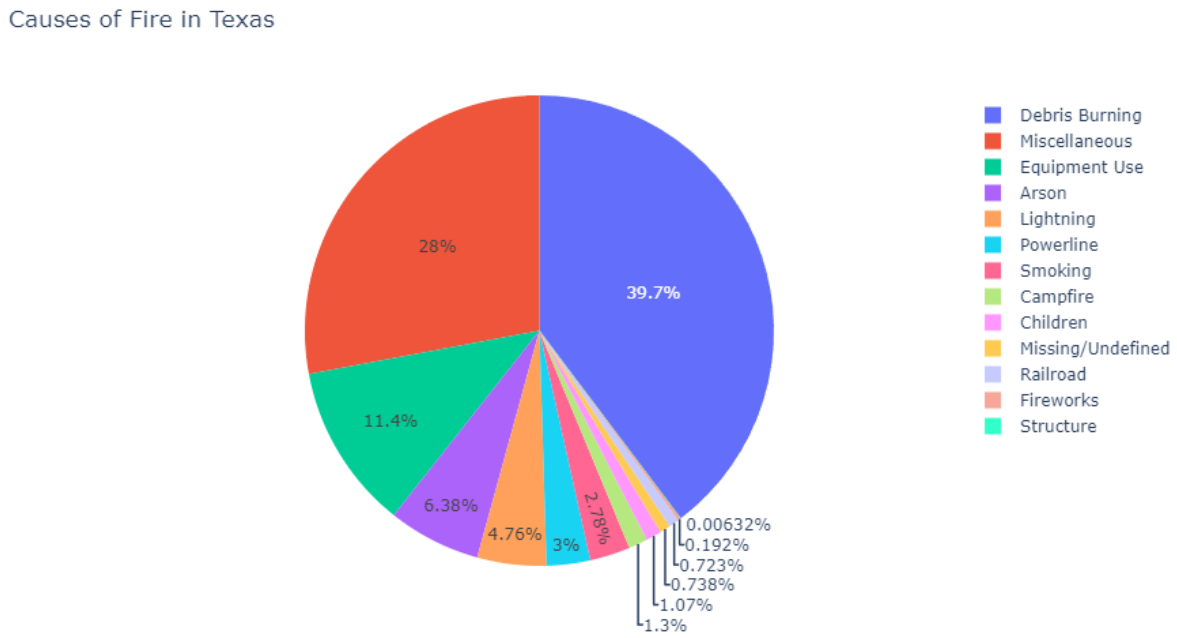
We can see that most fires occurred in California, while the state with the least amount of fires is Delaware.

Causes of Fire in California



Causes of Fire in Georgia





When checking the top 3 states with the biggest amount of fires (California, Georgia and Texas in order):

- The most common cause for fire in California is "Miscellaneous", meaning fire of known cause that cannot be properly classified into any of the other standard causes of fires. One of the reasons can be that this is a state with mostly very high temperatures.
- Although miscellaneous fires are still common in Georgia and Texas, they are not as common as fires that were caused by debris burning, which is the most common cause for fires in both Georgia and Texas, with Texas having more than half of its fires caused by debris burning.

This gives us a general direction as for the cause of fire depending on which state the fire occurred.

Feature Engineering

After drooping irrelevant and redundant features, we are left with the following features

- **SOURCE_SYSTEM**
- **NWCG_REPORTING_AGENCY**
- **FIRE_YEAR**
- **DISCOVERY_DATE**
- **DISCOVERY_DOY**

- **DISCOVERY_TIME**
- **CONT_DAY**
- **CONT_DOY**
- **CONT_TIME**
- **FIRE_SIZE_CLASS**
- **LATITUDE**
- **LONGITUDE**
- **OWNER_DESCR**

Now we'll try to make new features that will help us better categorize the cause of the fire. Some features we used as is, and some we had to manipulate.
In some features the distribution of the target feature against a specific feature, generated insights on the feature contribution to the model's ability to separate the data, generalizing better.

Fire Duration Feature

A feature that we thought would be interesting to create is the fire's duration in hours. We used 4 features to create the fire duration: **CONT_DATE, CONT_TIME, DISCOVERY_DATE, DISCOVERY_TIME**. First, we noticed that there are 37713 rows (around 50% of the data) which contain similar values for the containment date and time and discovery date and time. Since this does not make sense, we've decided to erase these rows. Secondly, we had to convert the values of the date features to the julian calendar, and then combine the date and time together for both the discovery and containment feature to create the date format: **y-m-d hh:mm**. Finally, we calculated the difference between the containment date and the discovery date in hours.

We also had to impute the null values. The imputation we chose is:

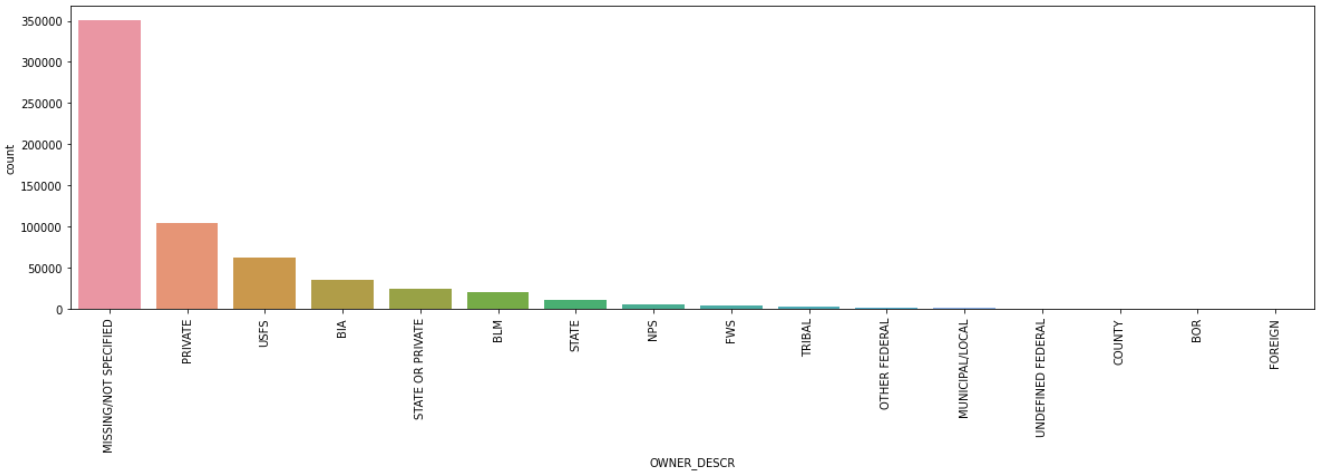
1. Calculate the average fire duration for each group of fire size. Let: 'avg' be the number calculated for a specific row.
2. Fix a number: epsilon = avg/4
3. Choose a value uniformly in the range [avg-epsilon, avg+epsilon] to replace the null value.

Now that we have this new feature we dropped the 4 features that we used.

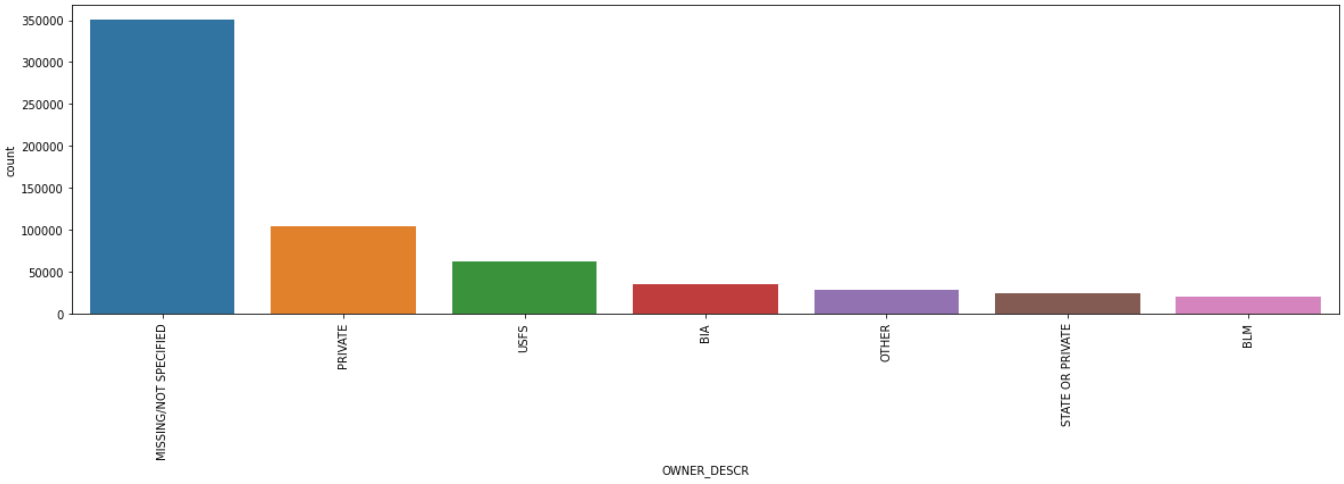
NOTE: The imputation was done **after** the split of the data to train and test sets in order to avoid data leakage, meaning the imputation was done separately on the train set and on the test set.

Owner Description Feature

OWNER_DESCR is a categorical feature. After plotting its distribution:



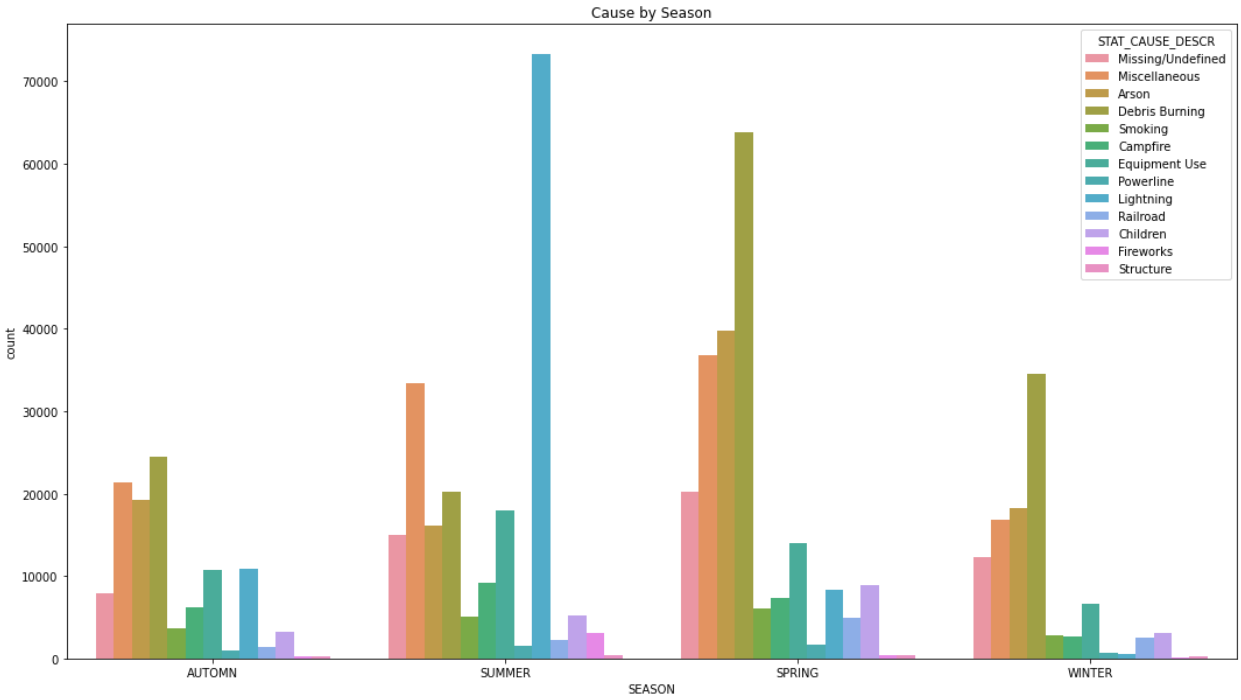
We can see that there are much less rows with the values: **STATE, NPS, FWS, TRIBAL, OTHER FEDERAL, MUNICIPAL/LOCAL, UNDEFINED FEDERAL, COUNTY, BOR, FOREIGN** than the other values. That means those categories are quite rare, and can be classified under "**OTHER**". After binning:



Season Feature

Another feature we thought would be interesting to create is **SEASON**. We wanted to add this feature to check if we could better isolate the cause of the fire, for example maybe there are high temperatures during the summer, which can cause wildfires. Moreover, winter is a season with many holidays (Christmas, Halloween)

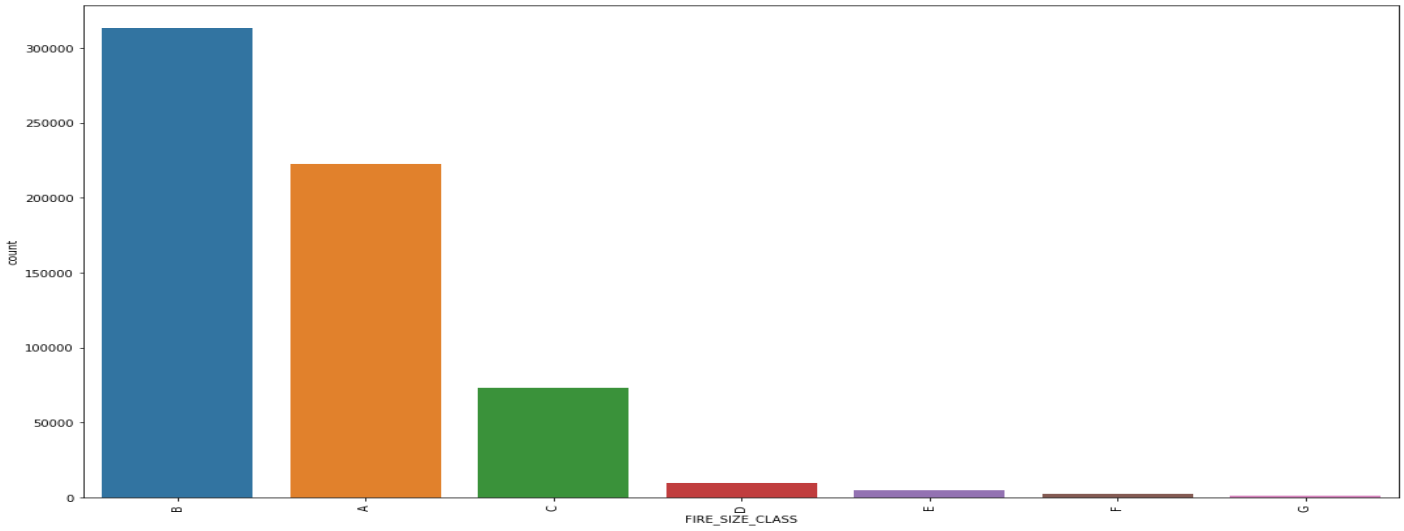
which might contribute to an increase in fire incidents. Let's take a look at the distribution:



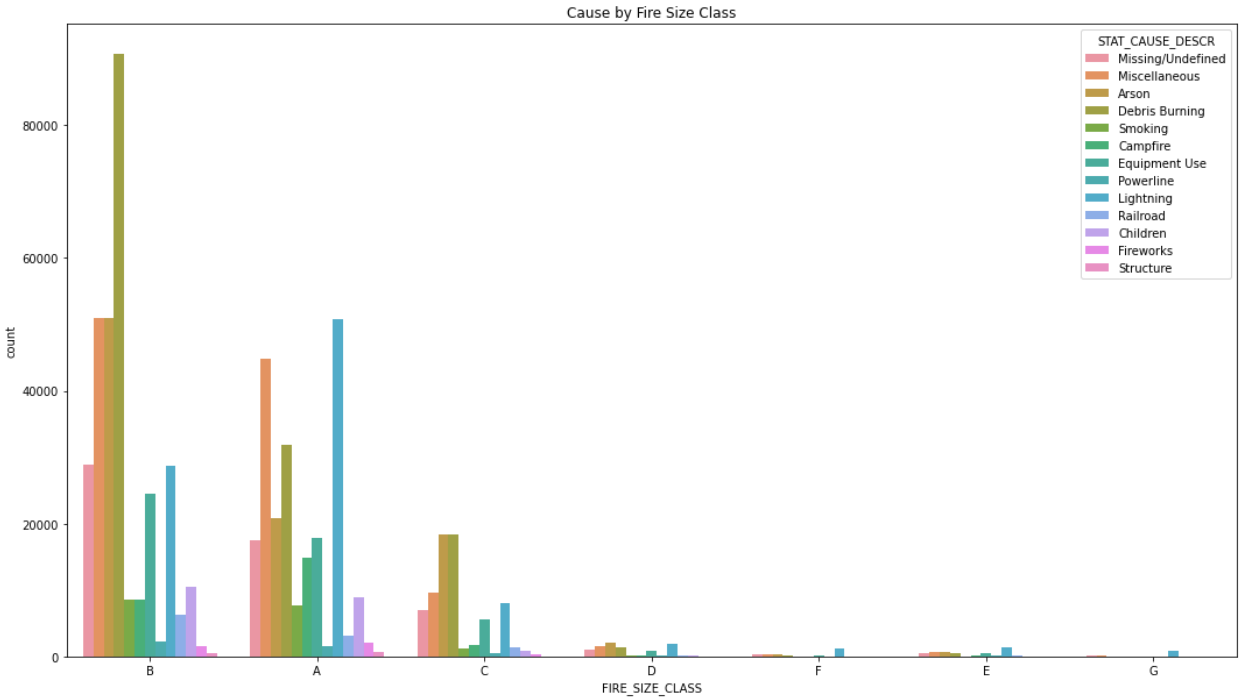
We can see, for example, that the absolute majority of fires that broke out due to lightning occurred in the summer, this is logical since it is known that in the summer there is extremely high moisture and high temperatures, which contributes directly to the appearance of lightning storms which set fire to the already dry vegetation.

Fire Size Class Feature

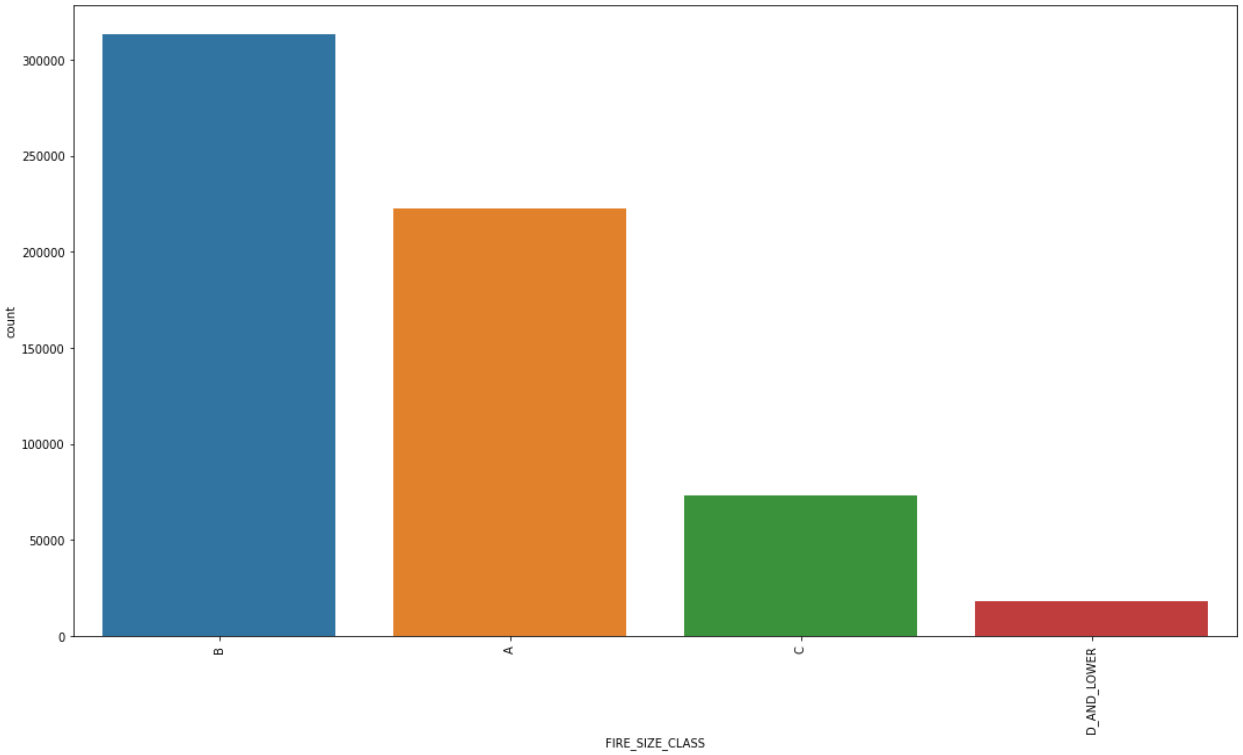
FIRE_SIZE_CLASS is a categorical feature. After plotting its distribution:



We can also see the distribution of those categories under each cause of fire:

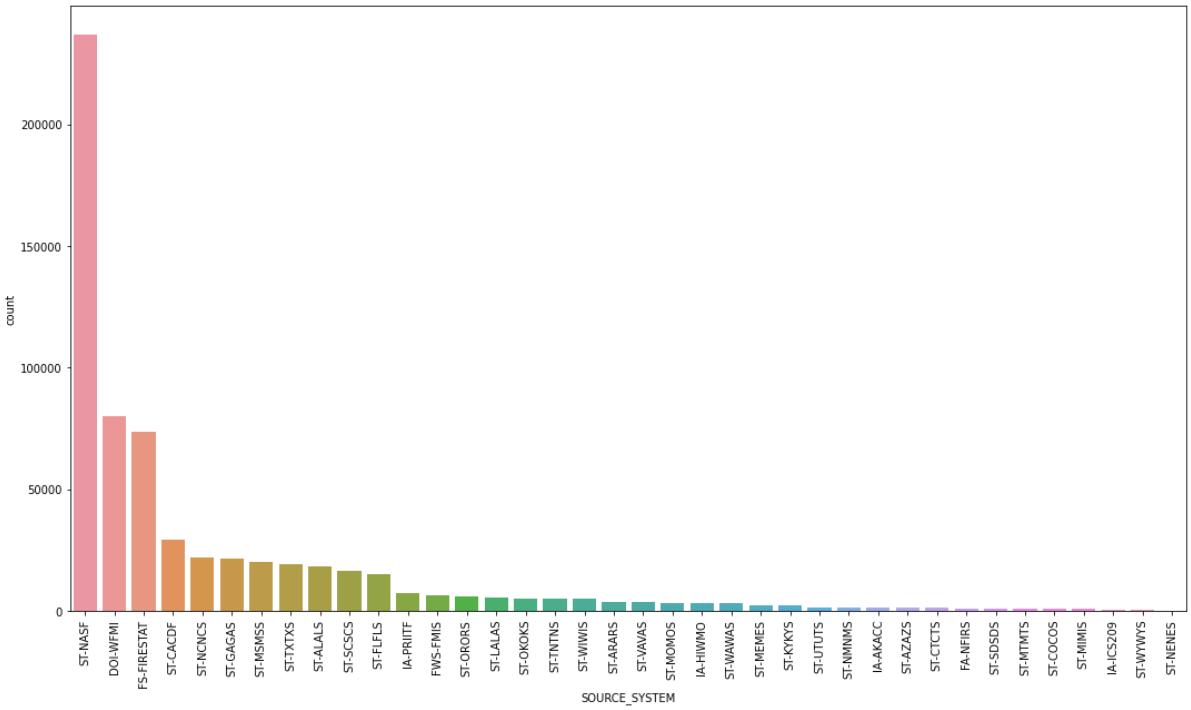


Interesting insight is the majority of fires sized "**A"** caused by lightning and **"B"** sized majority caused by arson. We can see that rows with fire size class: **D, E, F, G** are quite rare. We binned those categories and classified them under "**D_AND_LOWER**". After binning:

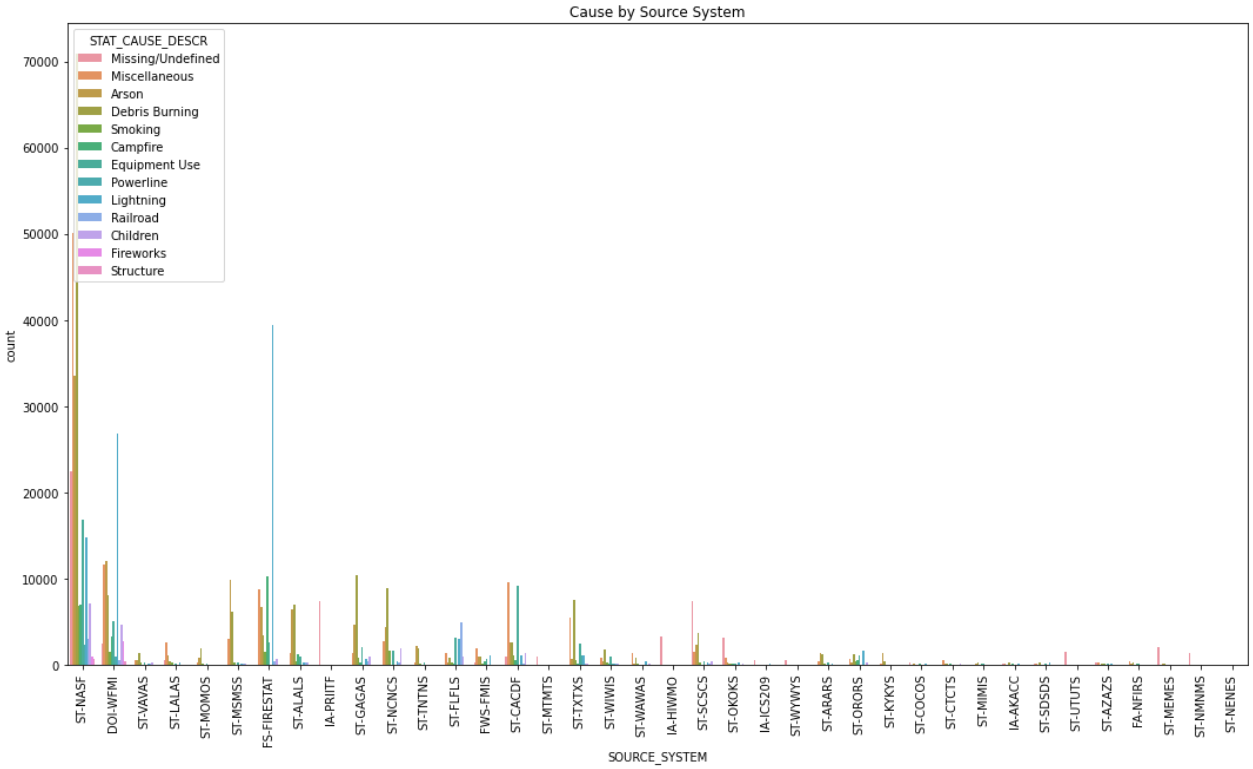


SOURCE SYSTEM Feature

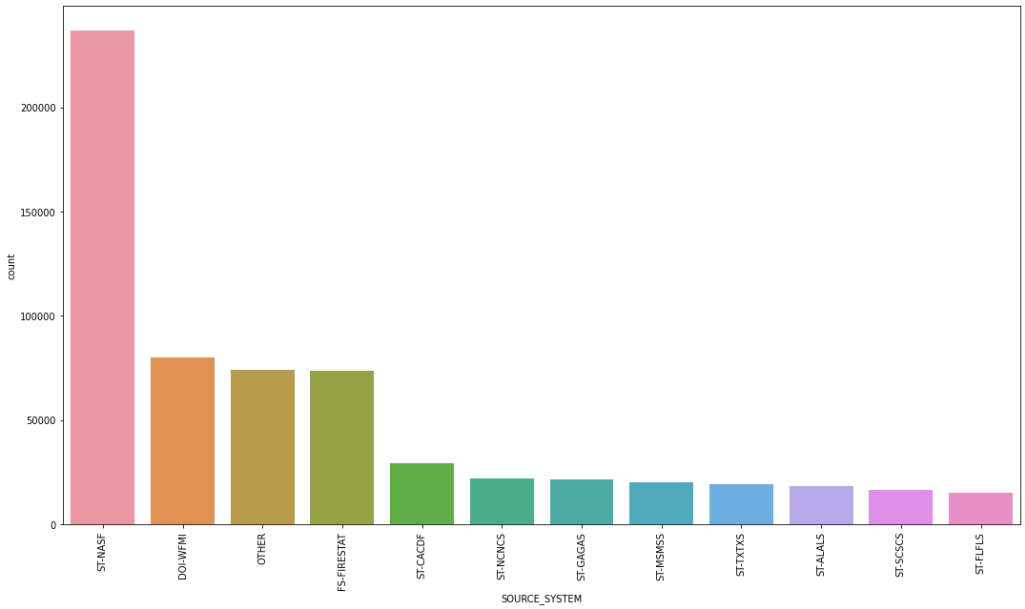
SOURCE_SYSTEM is a categorical feature. After plotting its distribution:



We can also see the distribution of those categories under each cause of fire:

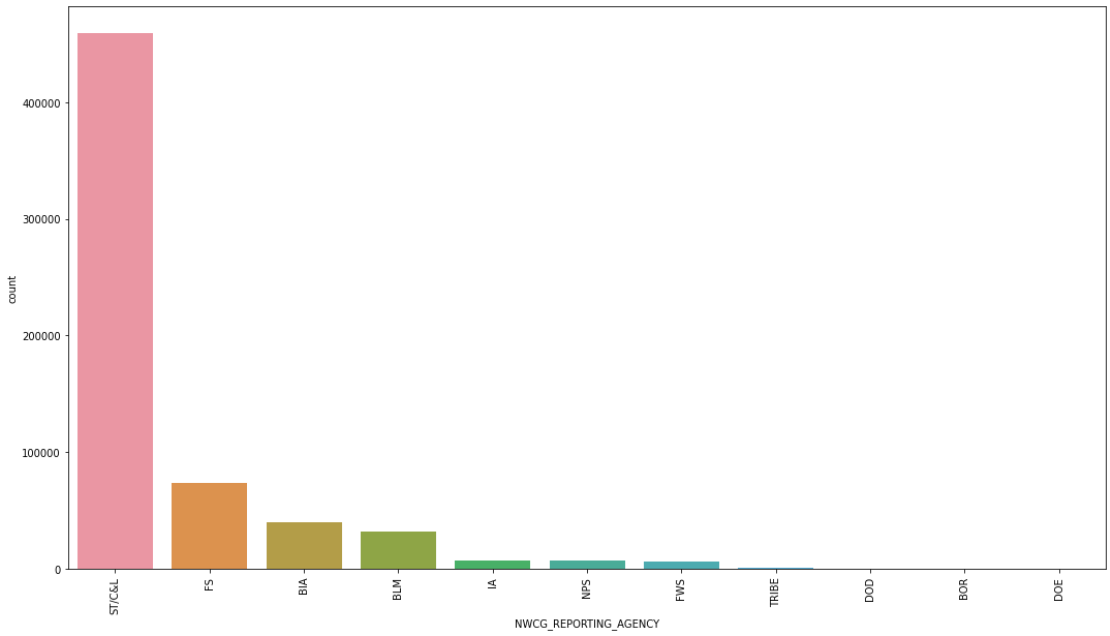


We can see that there are parameters which are quite rare and furthermore do not give us and significant correlation with any of the fire causes, so we decided to use binning where “OTHER” contain all the parameters with the small frequencies in the dataset:

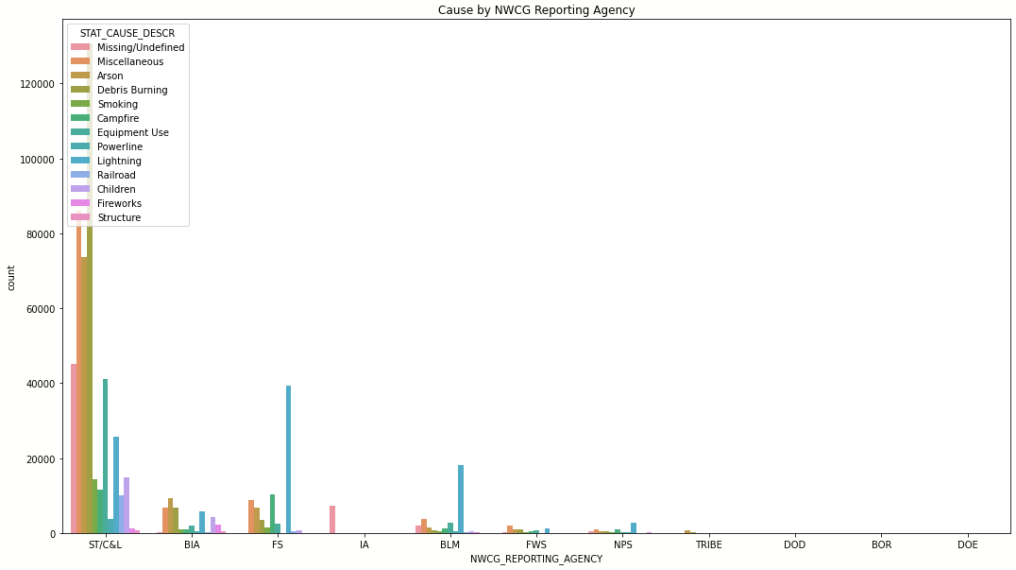


NWCG REPORTING AGENCY Feature

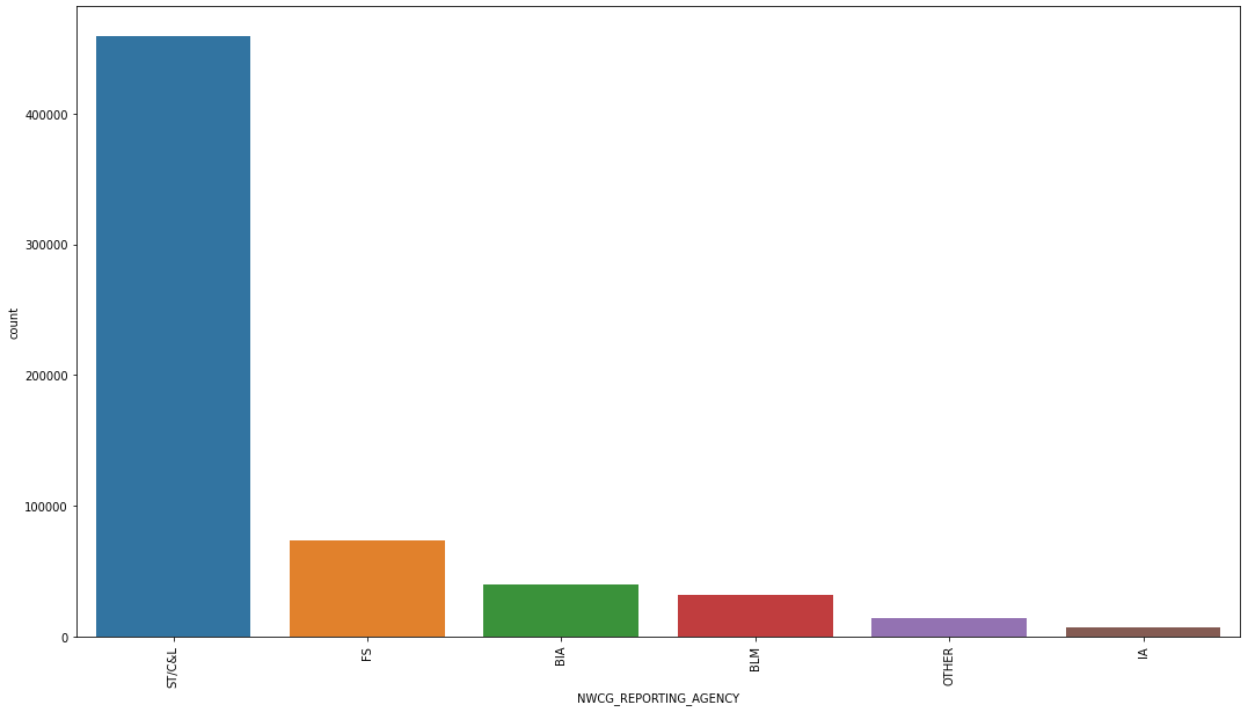
NWGC_REPORTING_AGENCY is a categorical feature. After plotting its distribution:



We can also see the distribution of those categories under each cause of fire:



We can see that there are parameters which are quite rare and furthermore do not give us and significant correlation with any of the fire causes, so we decided to use binning where “OTHER” contain all the parameters with the small frequencies in the dataset:



Dummy Variables

Our categorical features are **OWNER_DESCR, FIRE_SIZE_CLASS**, and **SEASON** so we've decided to convert these variables to dummy variables.

Failed Experiments

State Feature

The STATE feature possesses high cardinality - there are 50 states in the USA, meaning 50 different categories, and creating dummies for each one slowed our performance drastically.
Therefore, we had to think of different methods to deal with this feature.

The first method which deals with this is to calculate a simple aggregated value per group. This will create only 1 feature instead of additional 50.
In our case, we calculated the percentage of fire incidents per state and used that as our new feature.

The second method is to divide the states into 4 regions by their geographic location. This way we will only need additional 4 columns (after dropping the STATE column).

Both methods did not give better results in terms of f1 score, so we've decided to drop this feature.

Also, using the longitude and latitude features gives us enough information about the location of the fires.

Time of the Day

Another feature we thought will improve our model is a feature for time of the day: we used the DISCOVERY_TIME features, and created 4 categories according to the hours of the day: Morning, Evening, Afternoon and Night, and then created dummies. This did not improve our model in terms of f1 score.

Model Fitting

The main models we checked after researching on models which handle big imbalanced datasets, were:

- Random Forest Classifier
- Histogram Based Gradient Boosting Classifier

We chose the HistGradientBoosting model over the Random Forest model because it has been shown to produce better predictions in many cases.
HistGradientBoosting is able to handle high-dimensional data more efficiently and can often achieve better accuracy on complex datasets.
Indeed, the Histogram Based Gradient Boosting Model performed better with the provided dataset.

Histogram based gradient boosting model

HistGradientBoosting is a machine learning algorithm that is used for gradient boosting on large datasets.

The algorithm is specifically designed to handle sparse and high-dimensional data efficiently by using histogram-based algorithms for splitting features.

The most important parameters for HistGradientBoosting are the learning rate, the maximum number of iterations, and the maximum depth of each tree. Other important parameters include the minimum number of samples required to split a node and the minimum number of samples required in each leaf node.

One advantage of using HistGradientBoosting is that it can handle missing values in the input data, which is a common issue in many real-world datasets.

One more advantage of this model is its internal implementation which includes sampling balanced data at each iteration (The model divides the data into buckets), this is only efficient when class_weights parameter is defined (explanation about this parameter, at the end of the report).

Hyperparameter Optimization

The hyperparameter optimization method we chose is Bayesian Optimization, and was done with “optuna” library on the following hyper-parameters:

- **max_depth**
- **max_leaf_nodes**
- **min_samples_leaf**
- **l2_regularization**
- **learning_rate**

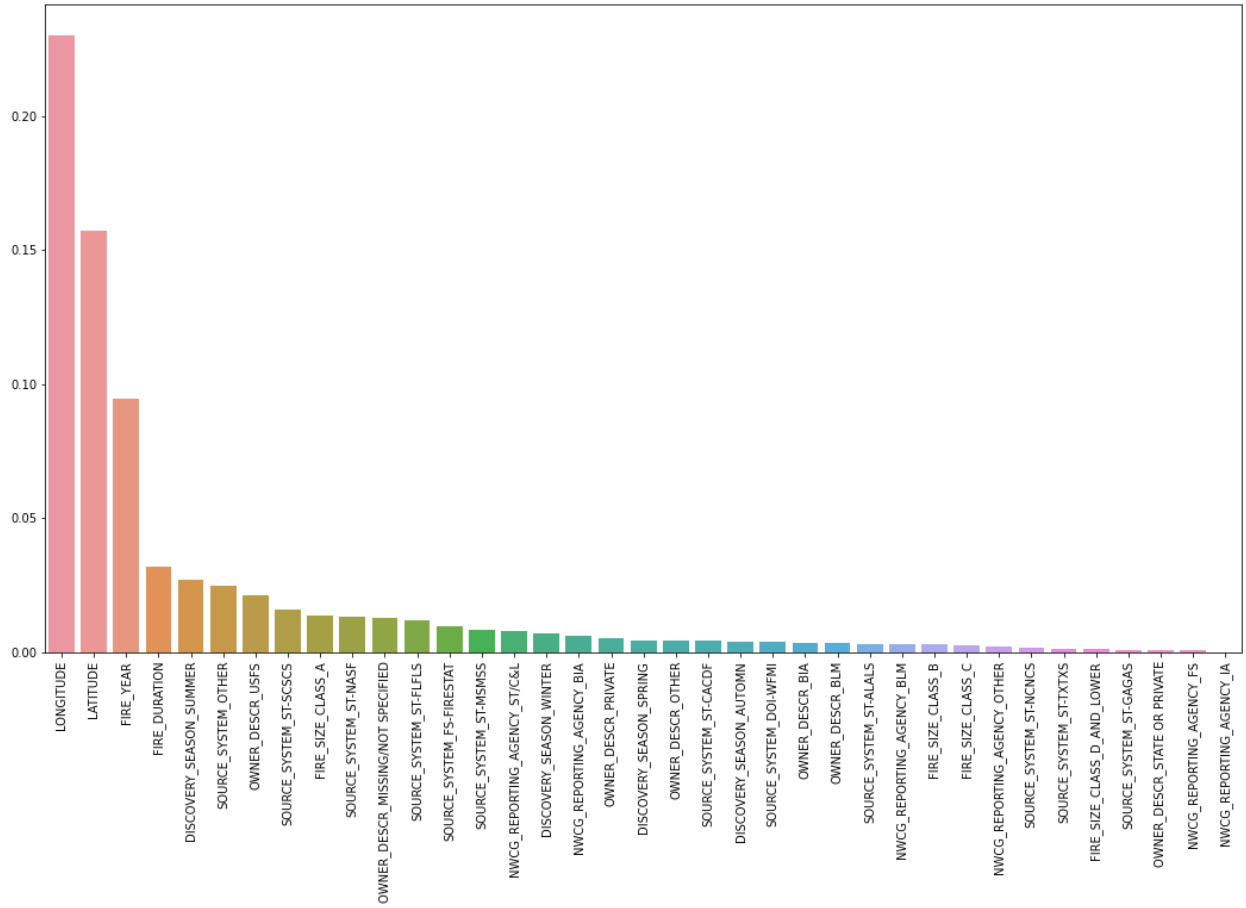
We ran a few studies with different ranges for those parameters. In each trial a different model was fitted, afterwards cross-validation was done on the fitted model according to the Stratified K-Fold method. The reason for choosing this type of cross-

validation is due to our imbalanced data: the Stratified K-Fold method stratifies the target classes, meaning that both the train and the validation sets will have an equal proportion of all classes.
The trial's output was defined to be the average score of the model after the "cv" process.
During the optimization we saw that a high score on the train and validation set doesn't mean at all the model will perform well on the test set. The reason was that the more complex models (e.g: high depth, large number of iterations) made the model too complex and resulted in overfitting.

Feature Importance

The method we used for feature importance is **permutation importance**. The reason for choosing this method is because it works for any type of model (unlike the **mean decrease in impurity** method) and is not biased. Moreover, the results of this method are easy to understand.

Our results:



- When we used the first 30 and 20 features, the weighted average f1-score did not change, but the weight average precision was slightly greater (changed

from 0.53 to 0.54). This does not necessarily mean the model is better - a better model will increase the recall and decrease the precision. Here, the precision increased but there might have been a change in recall (which probably decreased) but we don't see it because the numbers shown in the classification report only show 2 digits after the decimal point. Since we can't be sure, and the difference is very small either way, we don't consider these models to be better.

- When we used the 10 best features, the weighted average f1-score got smaller (from 0.53 to 0.52), hence it is not a better model.

Following these conclusions, we've decided to go with the full model (choosing all features).

Failed Experiments

Oversampling and Undersampling Techniques

As our data is highly imbalanced, we searched for ways to deal with such data and found that the common approaches were:

SMOTE (Synthetic Minority Oversampling Technique)

This approach oversamples the minority classes, and is applied on the training data only, hence the model trains on all of the classes. In general, the algorithm is:

1. Select random data from the minority class.
2. Calculate the Euclidean distance between the random data and its k nearest neighbors.
3. Multiply the difference with a random number between 0 and 1. Then, add the result to the minority class as a synthetic sample.
4. Repeat the procedure until the expected proportion of minority class is met.

After applying this technique over the training data, the weighted average f1-score was 0.51 on the test data, giving a lower result, although the model did succeed to predict some of the minority classes better in terms of the recall values:

Without SMOTE, the recall and precision values for the following minority classes were:

	precision	recall
Powerline:	0.17	0.04
Railroad:	0.43	0.39

Smoking:	0.19	0.01
Structure:	0.13	0.06

While with SMOTE, the precision for each class decreased and the recall increased:

	precision	recall
Powerline:	0.08	0.31
Railroad:	0.31	0.48
Smoking:	0.13	0.07
Structure:	0.05	0.29

Though the weighted average f1 score decreased since this came at the expense of the prediction for the majority classes.

Tomek-Links

Tomek Links is a method of under-sampling. It can be used to find desired samples of data from the majority class that is having the lowest Euclidean distance with the minority class data, and then remove it. This method gave similar results to the SMOTE results, and did not improve the model's performance on the test data.

SMOTE-Tomek-Links

As the name suggests, this method combines both of the above methods.
Due to very high runtime (the code did not stop running), we've decided to drop this method.

Class Weights

This is the parameter in this model which relates to how the model handles on imbalanced datasets. The weights associated with classes in the form {class_label : weight} . If not given, all classes are supposed to have one weight. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)).

Random Forest Model

We followed the same methodology as with the HistGradientBoosting model, but with no success in terms of f1-score.

Summary

Our final model is the Histogram-based Gradient Boosting Classification Tree, with the default parameters:

- **random_state**: 50
- **verbose**: 2
- **loss**: categorical_crossentropy

and the optimized parameters:

- **max_depth**: 13
- **max_leaf_nodes**: 25
- **min_samples_leaf**: 26
- **l2_regularization**: 0.09665996246059405
- **learning_rate**: 0.17822917024415935

(The optimized parameters might vary according to the sample of the data and the split for train and test sets).

The highest weighted average f1-score we achieved: **0.53**.