Applied Machine Learning
Group 21 Final Project Report

Xinyu He (xh2562), Dieter Joubert (dj2574), Ritvik Khandelwal (rk3213), Ethan Tucker (eht2122), Keli Wang (kw3015)

Introduction

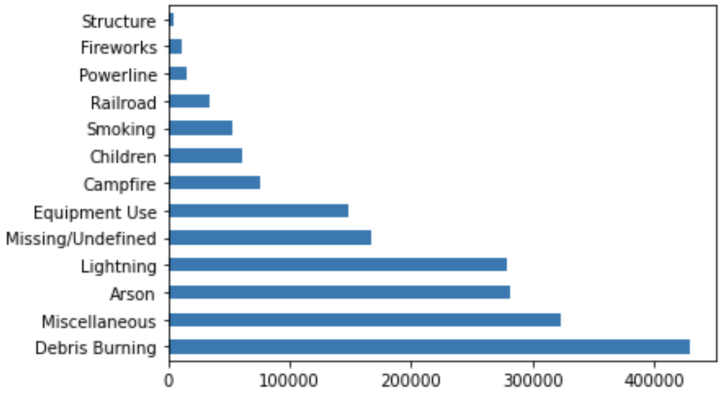
Wildfires have increasingly become a major concern for many urban and rural areas in the United States, with severe health, economic and social consequences. It's possible that global warming is contributing to the frequency and intensity of wildfires, with a number of studies showing changes to the features of wildfires, including increases in season length, frequency and burned area. We investigated the causes of wildfires and predicted the cause based upon a multitude of other features.

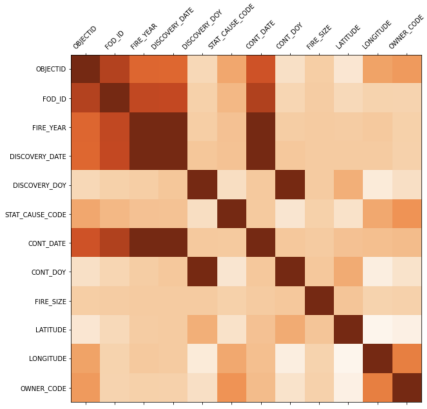
Dataset

We used a dataset from kaggle called "1.88 Million US Wildfires: 24 years of geo-referenced wildfire records." [1] This data spans the years 1992 to 2015. The dataset includes 51 features for each recorded wildfire. Some notable features include final fire size, longitude, latitude, hours spent in containing the fires, fire year and discovery month. The records were collected from the reporting systems of federal, state, and local fire organizations.

Data Exploration

Looking at the data, we find that out of 51 features, all features have no missing values except three columns DISCOVERY_TIME, CONT_DATE and CONT_TIME. The number of missing values they have is 882638, 891531 and 972553 respectively. There exists a highly correlated pair of features DISCOVERY_DATE and CONT_DATE between which the correlation is larger than 0.9.





Then we visualized the data and obtained some interesting findings regarding different aspects of wildfires. For instance, the burned area of wildfires tends to be larger in spring and winter, and smaller in summer. Debris burning is responsible for a vast majority of wildfires. When it comes to the relationship between cause of fire and fire size, arson results in the largest fire size on average, while campfire usually results in the smallest fire. Slightly more wildfires occurred during weekends and most of the wildfires were contained within 7 hours or 1.5 days after discovery. Finally, using a kernel density plot, we visualized the occurrences of each fire cause on a map of the United States, to understand the relationship between location and cause. This plot was compared with the maps of forest density shown below to further elucidate reasons for a spatial dependence to the different causes (see "Additional Data Exploration Visualizations" at end of document).

Methods and Algorithms

We used various machine learning methods to analyze the dataset in order to answer the question of what the cause of wildfires are.

1. Multinomial Logistic Regression

As a first, simple model, we attempted to use multinomial logistic regression to fit the data. Results with multinomial logistic regression, after playing around with hyperparameters, were far from satisfactory. Long training times also contributed to making this look like a candidate method which would not be a great match for this problem.

Our dataset is fairly high-dimensional, and logistic regression is well-known to overfit on training data given high-dimensional data, explaining our low accuracy. We attempted to pare-down the data to columns which looked like they might be of high-importance and not correlated with each other, but accuracy didn't improve as much as we hoped.

2. Support-vector Machines

We trained two different support vector machine implementations on a sample of training data. A huge drawback of the SVM was that, with the size of the training dataset and the number of features (even after dropping several columns), the SVM took very long to train (4.5 hours using Google colab Pro).

The best testing accuracy achieved after experimenting with hyperparameters was 56.99% with a linear-kernel SVM. While a RBF-kernel SVM can theoretically at least perform as well as the linear version, initial results and experimentation with hyperparameters didn't leave us feeling very confident that the RBF-kernel SVM could perform much better.

3. Decision-tree

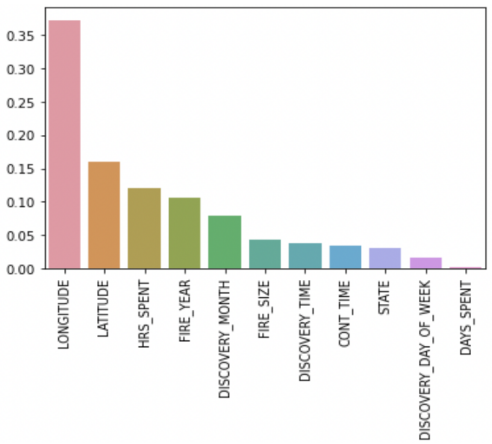
Moving further, we implemented tree techniques to perform multiclass classification. Trained multiple variants. Initially imputed the missing values with the median based on the distribution of the features and trained the model. Trained another model where we dropped all missing values and the latter model ended up performing equally good so dropped the missing values and tuned the classifier.

Due to the size of the dataset and resource limitations, the grid search took an indefinite amount of time so we resorted to random search to obtain optimal depth.

Performed random search to obtain optimal depth.

Additionally, we made a bar chart to infer feature importance. This helped us understand how the decision tree gave importance to different features. Resulted in dropping columns pertaining to Containment of fire.

The best decision tree classifier achieved an accuracy of 57.21% on the test set. We observed that the features Longitude, Latitude, Hours spent in containing the fires, fire year and discovery month were treated as few of the most important features while classifying the model.



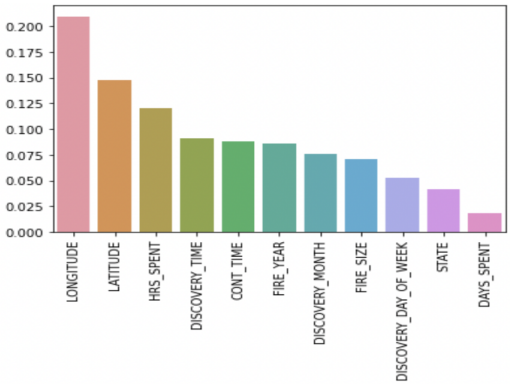
4. Random-forest

To generalize better, it was natural for us to shift to ensemble techniques. To this end, we used the Random Forest Classifier. Similar to what we did earlier, we first imputed the missing values with median and trained the model. Trained another model where we dropped all the missing values and the latter ended up performing equally good.

Due to the size of the dataset, the grid search took an indefinite amount of time so we resorted to random search to obtain optimal depth and n_estimators.

We made a bar chart to infer feature importance. This helped us understand how the classifier gave importance to different features. Resulted in dropping columns pertaining to Containment of fire.

After tuning, the Random Forest classifier achieved the best accuracy of 61.02%. Further inference can be found in the conclusion below. We observed that the features Longitude, Latitude, Hours spent and many others have been given much more importance as compared to the decision tree classifier.



5. Boosting

In this part we compare the performance of different boosting techniques on this multinomial classification problem. We tried two different kinds of boosting techniques: adaptive boosting and gradient boosting. For the gradient boosting technique, 3 different implementations GradientBoostingClassifier, HistGradientClassifier and XGBoost are used. When it comes to accuracy, we see that AdaBoosting performs worse than any of the gradient boosting techniques. It is possible that Adaboost does not work well for multinomial classification problems.

The running time of GradientBoostClassifier and XGBoost are significantly longer than other implementations. This may be related to the feature of XGBoost that it doesn't support categorical variables natively. The GradientBoostClassifier from sklearn doesn't use histogram to speed up splitting like what HistGradientClassifier does.

We also observe that recall on the test set is much lower than accuracy, indicating that a lot of false negatives occur. Many samples belonging to the minority class are predicted as other classes.

	Classifier	Training Time	Test Accuracy	Test Pecision	Test Recall
0	Gradient Boosting	72 min	0.5857	0.4555	0.3564
1	Hist Gradient Boosting	6.6 min	0.5928	0.4919	0.3641
2	XGBoost	53.1 min	0.6149	0.5503	0.3915
3	AdaBoost	5.4 min	0.4550	0.2319	0.1752

6. Neural Networks

As the dataset is very large, we figured that a simple feedforward neural network might be a good choice so we decided to leverage the power of the state-of-the-art tabular neural network architecture from the fastai library [2] as it has been benchmarked on many datasets previously.
We attained 49% accuracy with no hyperparameter tuning. We leveraged the power of cyclical learning rates [3] from the fast.ai library to obtain better accuracy.
We removed the missing values and experimented with learning rates (0.01, 0.001) and number of epochs and achieved best accuracy with learning rate = 0.001, number of epochs = 50, optimizer = Adam. Observed that imputing or keeping the missing values as it is gave worse performance (~51%)

```
TabularModel(
  (embeds): ModuleList(
    (0): Embedding(53, 15)
    (1): Embedding(13, 7)
    (2): Embedding(8, 5)
  )
  (emb_drop): Dropout(p=0.0, inplace=False)
  (bn_cont): BatchNorm1d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (layers): Sequential(
    (0): LinBnDrop(
      (0): Linear(in_features=35, out_features=200, bias=False)
      (1): ReLU(inplace=True)
      (2): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): LinBnDrop(
      (0): Linear(in_features=200, out_features=100, bias=False)
      (1): ReLU(inplace=True)
      (2): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (2): LinBnDrop(
      (0): Linear(in_features=100, out_features=13, bias=True)
    )
  )
)
```

Analysis

Models	**Best Test Accuracy (%)**
Multinomial Logistic Regression	31.08
Support Vector Machine	56.99
Decision Trees	57.21
Random Forest	61.02
Gradient Boosting	58.57
Histogram Gradient Boosting	59.28
XGBoost	61.49
AdaBoost	45.50
Neural Network	54.58

Conclusions and Future Scope

In the past, there has been significant EDA done on this dataset with little to no work on predictive analytics. As this was an unexplored area we decided to use Machine Learning techniques to predict causes of fire. Due to the volume of the data and lack of significant features, we obtained accuracies in the range of 50 - 65%. Perhaps, this dataset was collected with EDA and visualization in mind (also explains why there has been only EDA competition pertaining to this dataset with no Machine Learning).

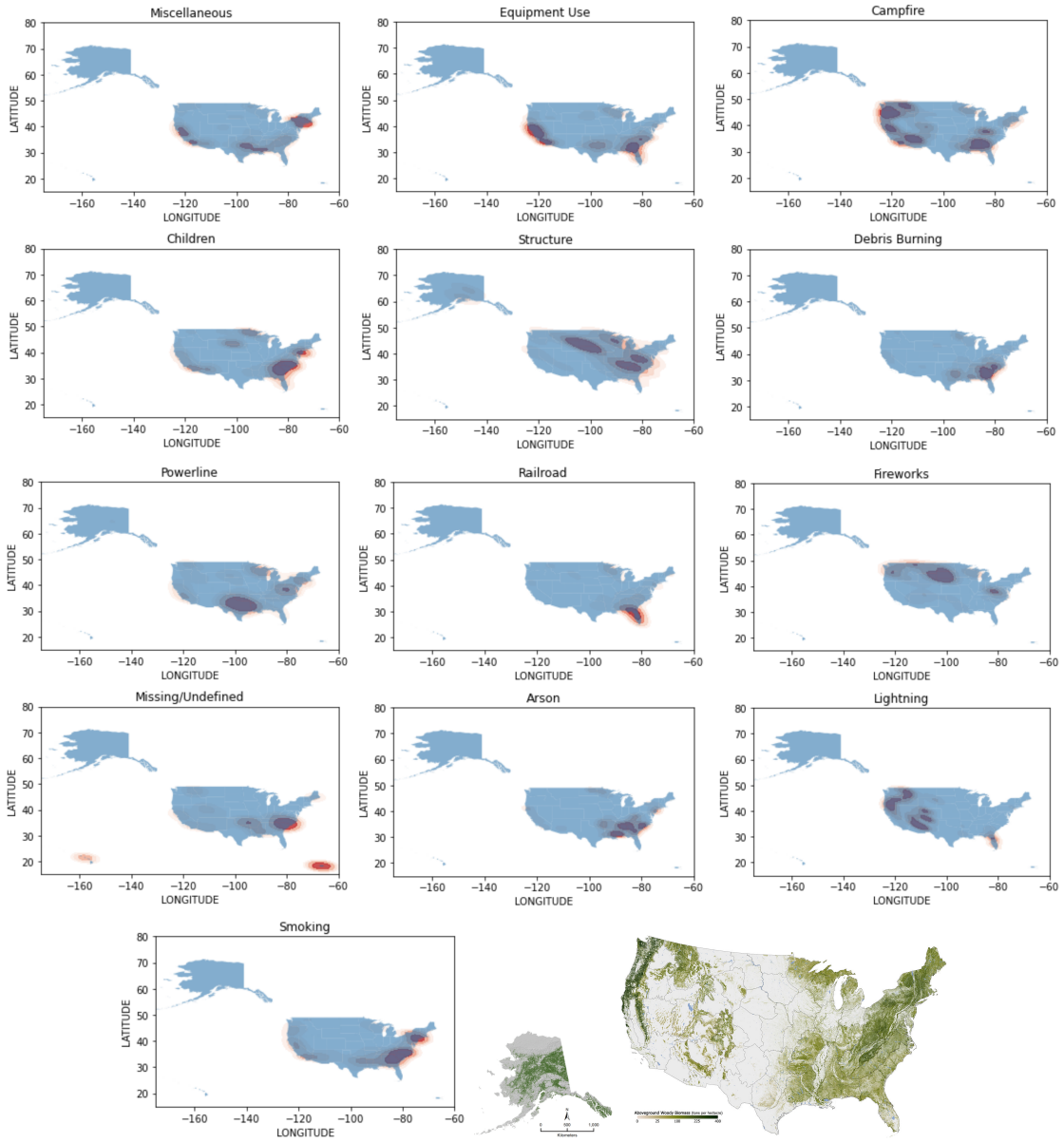
Despite this, we believe that we have obtained satisfactory results which will serve as a benchmark for future Machine Learning work on this dataset. XGBoost was our best performing model, which is probably because XGBoost is able to identify the minor classes and give more weight to the classes with low participation in the upcoming iteration. This feature makes it a good option for imbalance data we have here. The random forest classifier also performed very well. Looking at the feature importance graph above, it can be inferred that it outperformed other models because it has really generalized well to give significant importance to features that really matter, unlike the decision trees model which did not give as much importance to these features.

The neural networks probably did not perform as well because of the simple architecture. Perhaps, increasing the complexity of the architecture by designing a custom neural network architecture for this dataset, can be an interesting area of research. Support Vector Machines has given satisfactory results, however this model took several hours to train. Random Forest and Decision trees on

the other hand, were very quick. The ability to parallelize the training of random forests makes it all the more better from a deployment standpoint.

By implementing different variants of Decision trees, Random Forest, Neural Network, SVMs we have also concluded that removing the missing values has proven to be a better approach rather than imputing them or leaving them as it is. This is probably due to the fact that there are columns which have more than 45% of missing values and these columns are being treated as important features by our models so imputing these will only add more noise. Therefore, dropping the missing values has given us marginally better results.

Additional Data Exploration Visualizations



References

[1] https://www.kaggle.com/datasets/rtatman/188-million-us-wildfires
[2] https://docs.fast.ai/tabular.data.html
[3] https://arxiv.org/abs/1506.01186