Machine learning for wildfire prediction from meteorological data in California and Nevada, USA

Ari Fleischer[1], August Posch[2], and Oj Sindher[3]
Northeastern University[1,2,3]
fleischer.a@northeastern.edu[1], posch.au@northeastern.edu[2], sindher.o@northeastern.edu[3]

Abstract

This project used meteorological data to predict wildfire ignitions, temporally within 10-day windows, spatially within roughly 25 km of weather locations in California and Nevada, USA, using data from 2001-2010. Predictions were for ignitions in the same temporal window as the weather, which is not designed for short-term predictions based on current-moment weather, but is useful for creating a susceptibility map which informs long-term risks in the area. The best results were found using a 10-day temporal resolution, joining the fire data to the nearest weather station, imputing missing values using the mean, and scaling features using standard scaler. The best machine learning algorithm was a soft-voting classifier composed of a closed-form logistic regressor, a stochastic gradient descent classifier, and a random forest classifier and had an F1 score of 0.5138. This model with adjusting the threshold values achieved recall of 0.9880 and precision of 0.2020, which is better than results found in previous work optimized for Yunnan Province, China. Going forward, this project could be adjusted to optimize for short-term predictions, or could be applied to other regions of the globe.

Introduction

Wildfires kill people and destroy ecosystems if mismanaged. If we can predict fires within the next few days in a given area, then residents can be evacuated and firefighting crews can be sent in. If we can determine the long-term susceptibility of an area, then people can make informed decisions about whether to live there, and appropriate disaster contingency plans can be designed in advance.

Machine learning techniques have exploded in popularity in recent years, including in the realm of wildfires science. The number of publications per year in machine learning for wildfire science has increased fivefold from 2010 to 2020, and continues to steadily increase (Jain et al. 2020). Furthermore, machine learning practitioners can use state-of-the-art machine learning techniques to develop fire prediction systems without needing to have fire science expertise, thus saving more lives and environments.

Background

Most of the data loading, data preprocessing, and feature extraction used built-in functions from the NumPy, Python Data Analysis (pandas), and scikit-learn python libraries. The cfgrib library was used for reading meteorological data from GRIB files into python . The imbalance-learn library was used to test out undersampling and oversampling on the data. The machine learning algorithms used in this project were logistic regression, stochastic gradient descent (SGD), k-nearest-neighbors (KNN), decision trees, and a variety of ensemble, bagging, and boosting algorithms such as voting classifier, random forests, Adaptive Boosting (AdaBoost), and gradient boosting. All of these were run for classification. These algorithms were all run on the built-in scikit-learn implementations with the hyperparameters adjusted. Gradient boosting, however, was also run using the python library XGBoost which is an optimized distributed gradient boosting library.

Anomaly detection, also known as outlier detection, was used to identify points in the data set that deviate significantly from the rest of the data. The two anomaly detection algorithms used in the project were isolation forest and local outlier factor, which are both unsupervised machine learning techniques. Isolation forest works by an ensemble of decision trees where for each split point a random feature and random split point are chosen. An anomaly is detected by the average path length to isolate a certain sample over the trees, with a shorter path length indicating an anomaly. Local outlier factor measures the local density of samples compared to their k-nearest neighbors. The density of a sample is found by its distance to its nearest neighbors with the closer the sample the higher the density. An anomaly is detected through samples that have a much lower density than their nearest neighbors. The built-in implementation of both of these algorithms in scikit-learn was used.

Related Work

Cao, Wang, and Liu completed a paper, "Wildfire Susceptibility Assessment in Southern China: A Comparison of Multiple Methods" (Cao et al. 2017) in which they used logistic regression, probit regression, an artificial neural network, and a random forest to predict wildfire ignition, with the ultimate goal of creating a susceptibility map. Along the way they also investigated sample ratios of ignition and nonignition data to optimize model performance. They optimized their models for sensitivity (also known as true positive rate or recall) and specificity (also known as false positive rate), using an ROC curve.

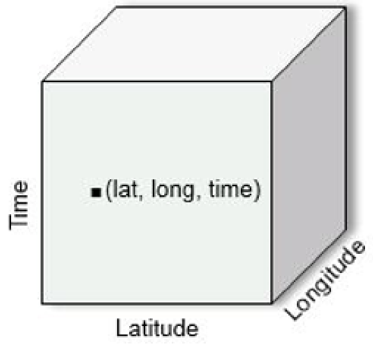
We are doing a few things differently from the Cao et al paper. We are choosing a different study area of California and Nevada, USA, chosen because the results are more relevant to the country we are currently living in, because wildfires are a large problem in these US states recently, and also because a wildfire ignitions dataset was more readily available to us for the US. We are using only CFSR weather data, no landform data or vegetation data that Cao et al used. Our weather dataset is also slightly different from theirs - both are from CFSR, but they contain different features; for example, we have no wind speed features, but we do have a cloud water feature that they did not have. We also investigate some data processing techniques, including effects of temporal bin sizes and missing values imputation. We are also trying more different techniques and algorithms, including feature engineering, k-nearest-neighbors, decision tree, support vector machine, adaboost, gradient boosting (including xgboost), and voting classifiers . We are also evaluating our models using the additional metrics of precision and F1-score, because we want to keep under control the rate of false alarms from our model.

There are other methods that could also be used to solve the problem of wildfire prediction, that we did not use in this project. We did not use probit regression because it was already tried by Cao et al. and found to perform very similarly to logistic regression, so we assume this holds true of our study area and trust that our logistic regression results account for this type of model. Support vector machines could be tried in greater detail but were not, because theoretically they would be less well-suited to our dataset, and we also have minimal experience constructing the kernels that are necessary to get an SVM to high performance. Similarly, other classifiers were not tried because they are theoretically ill-suited to our dataset; for example, our features are not very independent of one another, so Naive Bayes would be a poor choice. We could also try more different ensembles, and this is limited by the time we had available, as there were many combinations of classifiers and hyperparameters for each one.

Project Description

Datasets

There are two datasets utilized for this project. The first one is a subset of data collected by the Climate Forecast System Reanalysis (CFSR) system, which is a third generation reanalysis product (Saha et al. 2010). It is a global, high resolution, coupled atmosphere-ocean-land surface-sea ice system designed to provide the best estimate of the state of these coupled domains over years 1979-2011. This dataset can be requested in parts from the Research Data Archive (RDA), which is managed by the Data Engineering and Curation Section (DECS) of the Computational and Information Systems Laboratory (CISL) at the National Center for Atmospheric Research (NCAR) (UCAR, 2021). While requesting the data from RDA, one can specify the resolutions and also, subset the area and years to download the data for. The temporal and spatial resolution we selected for our data subset is every 6 hrs, and every 0.5 degree lat/long (~50km) respectively. This dataset is stored in General Regularly-distributed Information in Binary (GRIB) format, which is a file format for the storage and transport of gridded meteorological data, such as Numerical Weather Prediction model output. It stores the data in binary format but can be thought of as a 3D array for each parameter as shown in the figure below. So all the parameters have these three dimensional arrays, for example, you can get value for pressure for any location at a particular time.



GRIB format is a special way of storing the latitude, longitude, and time of each measurement.

Because this dataset stores multiple parameters, across decades, and for all locations on earth at such a high resolution level, this is big data. Therefore, due to limitations of resources and time, we chose to use California and Nevada weather data over the period 2001-2010. This covers latitudes between 32 and 42 degrees North, and longitudes between 125 and 114 degrees West. The reason for choosing this area is that these areas are often cited as having the highest wildfire risk among all the states of the USA (Moorcraft 2019). Data preprocessing was required to convert and join multiple GRIB format files into a dataframe which we could further process. The 3D parameters were converted into a 2D data structure with columns for latitude, longitude, and date-time.

NORTH AMERICA

The bounding box in the above image shows the study area - California and Nevada, USA - in the context of North America.

The initial weather data frame contained seven weather features: cloud water (kg/m2), relative humidity (%), total ozone (DU), geopotential height (gpm), precipitable water (kg/m2), pressure (pa), and GRIB parameter. (The GRIB parameter may not contain weather information, but we left it in the dataset in case it contained any useful information).

The second dataset is a spatial wildfire occurrence data for the United States, for period 1992-2015, present in the Forest Service Research Data Archive. It is a SQL database, and we selected year, day of year, latitude, and longitude for each ignition.

Data Transformations and Feature Extraction

Due to the nature of our data, the correct order to do some of the feature extraction was before we joined the weather data to our fire ignition target data. One of the techniques we investigated was what the ideal multiday binning strategy would be. For multiday bins of 3, 5, and 10 days, we extracted temporally-grouped summary features, as well as time-of-year features, and then joined the weather dataset to the fires dataset at the same temporal binning. For each multiday bin, we extracted the minimum, maximum, mean, and variance of cloud water, relative humidity, total ozone, geopotential height, precipitable water, pressure, and GRIB parameter. Because pressure had some missing values, we also extracted a count of missing pressure. For time-of-year features, we extracted day of year from our date-time, and did algebra to extract macro-season (first versus second half of the year) and month (12 months evenly divided - every 30.5 days). When a multiday bin overlapped multiple months or macro-seasons, we used the month or macro-season that contained the most days from that bin.

Once we had this multiday-binned, feature-extracted dataset, we were ready to join in our fire ignitions. To prepare the fires dataset we first rounded the fires latitude and longitude to the nearest half degree, which was the resolution of the weather stations. This ensured that each fire would be associated with the nearest weather location. To handle the temporal aspect, we gave each fire a key corresponding to the appropriate multiday bin based on its year and day of year, then grouped up to the level of the multiday bin and dropped any duplicates within that bin - this ensured that for each value of the index, there was at most one row. A 1 would mean that at least one fire occurred within the square of land surrounding this location, within this multiday window. Then we left-joined this fires dataset to our weather dataset based on latitude, longitude, and the appropriate multiday key, filling in any non-matches with a 0 in the fire column. A 0 means zero fires occurred within the square of land surrounding this location within this multiday window.

One way to think of the spatial aspect of the above join is that each weather location is the center of a rectangle. From that center, you would need to travel 0.25 degrees latitude north to reach the northern edge of the rectangle, 0.25 degrees latitude south to reach the southern edge of the rectangle, 0.25 degrees longitude west to reach the western edge of the rectangle, and 0.25 degrees longitude east to reach the eastern edge of the rectangle. Thus, this area is a 0.5 degree by 0.5 degree rectangle, with the weather location at the center. Any fires that occur within the rectangle are associated with the weather at the center of the rectangle.

Testing Effectiveness of Preprocessing Techniques

Using logistic regression, we tested different ways of approaching the preprocessing techniques of feature selection, missing values imputation, test-train split. We evaluated each of these using recall, precision, and F1 score. We ran logistic regression on each of our three datasets - 10-day binned, 5-day binned, and 3-day binned - and found that we achieved the best F1 score on the 10-day dataset. Note: the 10-day dataset had 176295 observations. We also ran logistic regression with many permutations of features to drop and found that it was always better to leave all the features in. Similarly, we found that dropping the missing-value pressure features performed similarly well to mean or median imputation, so we left the imputation method open to further tuning as we explored various classifiers. Finally, we considered the best way to make a train/test split. Due to our initial misunderstanding of how to make long-term predictions versus short-term predictions, we were treating this like a time series, short-term prediction problem, so we created our train/test split based on time: years 2001-2008 was the training set, and years 2009-2010 was the test set. Later on, we realized a random train/test split would be more appropriate for our purpose of preparing for a fire susceptibility map, because we were making predictions about fires within the same timeframe as our weather data. At this point we tested using a random train/test split instead, and it performed better. This is not surprising because weather data can be very different from year to year especially recently because of global warming. As a result, the data for 2009-2010 might have been very different from that of 2001-2008. While the random train/test split did perform slightly better on the basic classifiers, we ran out of time to run it on the more advanced classifiers and algorithms. To provide continuity with all our project's models, we are using our (unideal but serviceable) temporal split, where 2009-2010 were saved for testing. This is what we report on in our Results section below. For more comments on this, see "Future Directions" below.

Other preprocessing techniques were less clear cut as to the best approach. These included mean/median imputation, class imbalance techniques, and additional feature extraction. We explored these in various combinations with various classifier hyperparameters, discussed below.

We had a big class imbalance as the target had 147611 nonignitions and only 28684 ignitions (this is over a 5:1 ratio). For the class imbalance techniques, both oversampling and undersampling were tested on the 10-day dataset. Both were tested with ratios of 1:1, 1:2, 1:3, and 1:4 of the positive class to the negative class. Another way we dealt with class imbalance was by making the classifier class_weight hyperparameter “balanced” when possible.

Tuning Classifiers

Both logistic regression and SGD with loss = “log” were run on all datasets as a preliminarily classifier to figure out what dataset was the best. On the 10-day dataset, SGD was also run with different alpha values to see if adding regularization helped. Regularization didn’t help because overfitting was not a problem. Also, different KNN classifiers were tested using different distance metrics (cosine, euclidean, and manhattan) and the default and rule of thumb (sqrt(number of training points)) values for k. Traditional decision tree and random forest algorithms were also run on the 10-day dataset. Voting classifiers, both hard and soft voting, were used on the 10-day dataset. The classifiers used for the voting classifier were a combination of logistic regression, SGD classifier, and either decision tree or random forest.

As the boosting algorithms often outperform the basic algorithms given above, we tried a bunch of them, such as AdaBoost, GradientBoosting, and XGBoost algorithms, but they were found to be overfitting on the training dataset and hence, did not perform well in generalizing the results. As the classes were imbalanced in the data with 83:17, which is not normal, we thought of running a couple of anomaly detection algorithms, namely IsolationForest and LocalOutlierDetection, because they could detect wildfire ignition as anomaly in the weather data.

After running all these algorithms on a 10-day summary dataset, we decided to shorten our research to the best algorithm found so far, which was a voting classifier with three estimators, a logistic regression, a SGD classifier, and a RandomForest classifier. Thereon, to further enhance our results, we tried to supplement this best found voting classifier by adding the binned latitude and longitude variables. Initially, we did not include them thinking that they would increase the redundancy, but they were later found to be enhancing the overall results.

We were able to significantly enhance the classifier’s performance by adding polynomial features transformation to our set of features. The degree parameter of polynomial features was also tuned to give the best results. Lastly, to tune the classifier for higher recall, we adjusted the threshold value for decision function of the classifier.

Empirical Results

Temporal Binning Strategy

We chose a logistic regression classifier as the base model to test how summarizing the features by different numbers of days affects the results of a base classifier. For each logistic regression, we used mean imputation and used the hyperparameter class_weight = “balanced”. The results found are summarized in the below table, and based on these we decided to move forward with the 10-day summary dataset.

Bin size of dataset	Test set precision	Test set recall	Test set F1 score
3-day	0.1252	0.7055	0.2127
5-day	0.1751	0.6980	0.2799
10-day	0.2520	0.7078	0.3717

Missing Values Strategy

We tested multiple ways of dealing with missing values. Below are all logistic regressions using the 10-day dataset, and the hyperparameter class_weight = “balanced”.

Missing values strategy	Test set precision	Test set recall	Test set F1 score
Remove features with any missing values	0.2533	0.6986	0.3718
Impute with median	0.2526	0.7062	0.3721
Impute with mean	0.2520	0.7078	0.3717

Because the F1 score was so close whether we removed missing-value features, imputed with mean, or imputed with median, we continued trying all of these strategies in our pipelines as we tested new classifiers. Due to how classifiers other than logistic regression behave, imputing with mean ended up being the best, so below results will be given for mean imputation.

Class Imbalance Strategy

All of the below are logistic regressions (on the 10-day dataset, using mean imputation) run using different strategies for dealing with class imbalance. The first two strategies listed did not use the imblearn package. The other eight strategies used imblearn for random oversampling and random undersampling and then fed into a logistic regression with no class weighting.

Class imbalance strategy	Test set precision	Test set recall	Test set F1 score
class_weight = None (i.e., no class weighting)	0.4073	0.0357	0.0657
class_weight = "balanced"	0.2520	0.7078	0.3717
Oversampling to 1:4	0.3973	0.0993	0.1589
Oversampling to 1:3	0.3885	0.2246	0.2846
Oversampling to 1:2	0.3309	0.4062	0.3647
Oversampling to 1:1	0.2539	0.7024	0.3729
Undersampling to 1:4	0.3918	0.0959	0.1541
Undersampling to 1:3	0.3889	0.2272	0.2868
Undersampling to 1:2	0.3304	0.4086	0.3653
Undersampling to 1:1	0.2529	0.7044	0.3722

No strategy outperformed class_weight = "balanced" in the parameters available for scikit-learn's LogisticRegression.

Note: when we used imblearn oversampling or undersampling followed by class_weights = "balanced", our results were comparable to simply using class_weight = "balanced", so we did not show them here.

Train/Test Split Strategy

We tested out the 10-day dataset with mean imputation and class_weight = "balanced" in a logistic regression, in order to test a random train/test split versus a temporal train/test split. The random train/test split had a test set size of 25%. The temporal split used years 2001-2008 as training data and years 2009-2010 as test data.

Train/test split strategy	Test set precision	Test set recall	Test set F1 score
Temporal: train on 2001-2008, test on 2009-2010	0.2520	0.7078	0.3717
Random train/test split	0.2818	0.7174	0.4046

In this project we went forward with the temporal split, even though this performs worse and is not theoretically optimal. This was due to a misunderstanding and the limited time we had available. For more information please see the section above, "Testing Effectiveness of Preprocessing Techniques".

Comparing Classifiers

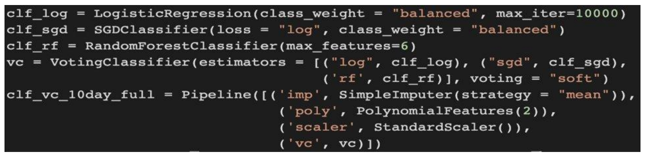
We tried numerous sklearn classification algorithms on the 10-day dataset and tuned their hyperparameters using trial-and-error methods. We also tried anomaly detection algorithms, explained earlier, because the class imbalance was quite high, but they were found to be highly underfitting, so we will not pursue them further.

The below table shows the results on the test set of the 10-day dataset for the best parameter settings we found for each of the algorithms. For example, the first voting classifier (LR+SGD+DT) was found to give better results with “hard” voting, whereas, the second voting classifier (LR+SGD+RF) gave gave higher F1 score value on test set with “soft” voting setting, and KNN resulted best when the weights parameter was set to "distance", and metric was set to "cosine". Similarly, the best parameters for SGD were “log” loss and “balanced” weights. The Decision Tree and other boosting algorithms results correspond to their default settings in sklearn.

Algorithm	Test set precision	Test set recall	Test set F1 score
Logistic Regression (LR) without class balancing	0.3979	0.0985	0.1580
Logistic Regression with class_weight = "balanced"	0.2422	0.7321	0.3639
SGD Classifier	0.2584	0.6608	0.3715
KNN	0.3138	0.2420	0.2733
Decision Tree Classifier	0.2425	0.3753	0.2946
Voting Classifier (LR+SGD+DT)	0.2894	0.5930	0.3889
SVM	0.5246	0.0064	0.0127
Gradient Boosting Classifier	0.5038	0.1710	0.2553
Random Forest Classifier (RF)	0.4649	0.2125	0.2917
AdaBoost Classifier	0.4333	0.1473	0.2199
XGBoost Classifier	0.5455	0.1286	0.2080
Voting Classifier (LR+SGD+RF)	0.3060	0.5956	0.4043

We concluded that the voting classifier with three estimators, namely, a logistic classifier, a stochastic gradient descent classifier, and a random forest, was performing best for our 10-day dataset. This classifier was used for further tuning of the results.

It is to be noted that the results in the table above do not have latitude and longitude as predictor features, which we later decided to include in our final model, as they were found to increase the F1 score on the test set. We also tuned the number of bins for latitude and longitude while binning them, finding that 15 bins for each results in the best results. To further enhance the results we utilized the degree 2 polynomial feature transformation too. Overall a pipeline was built which passed the features in 10-day dataset through *sklearn::SimpleImputer(strategy="mean"),* imputing the missing pressure values, then through *sklearn::PolynomialFeatures(degree=2)*, creating higher degree features and mixing existing features to create complex features as well, then scaling all the features using *sklearn::StandardScaler()*, and finally passing it from the best Voting Classifier discussed earlier. The results from this pipeline are given below.



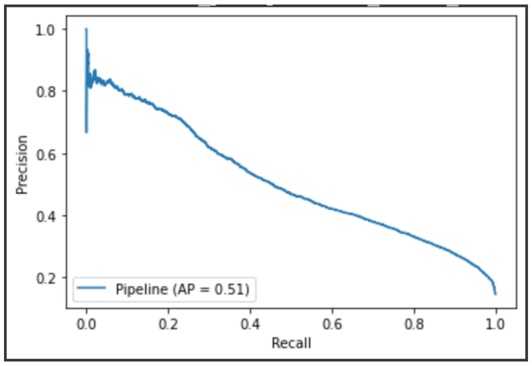
```
clf_log = LogisticRegression(class_weight = "balanced", max_iter=10000)
clf_sgd = SGDClassifier(loss = "log", class_weight = "balanced")
clf_rf = RandomForestClassifier(max_features=6)
vc = VotingClassifier(estimators = [("log", clf_log), ("sgd", clf_sgd),
                                    ('rf', clf_rf)], voting = "soft")
clf_vc_10day_full = Pipeline([('imp', SimpleImputer(strategy = "mean")),
                              ('poly', PolynomialFeatures(2)),
                              ('scaler', StandardScaler()),
                              ('vc', vc)])
```

Code snippet showing the pipeline for the best classifier found.

Metric	Training set	Test set
Accuracy	0.8875	0.8018
Precision	0.6123	0.3932
Recall	0.9012	0.7409
F1 score	0.7292	0.5138

Scores of the best classifier after the pipeline on the training and test sets.

These results show that the classifier is still a bit overfitting as its performance drops on the test set compared to its performance of the training set, but with this classifier we were able to achieve the highest F1 score on the test set. Therefore, this classifier was used to further tune the results by changing the threshold values. The below infographic shows the tradeoff between precision and recall for the test set.



Precision-recall tradeoff graph for the best classifier, showing the roughly linear tradeoff between the two.

As we would want our classifier to correctly classify all the wildfires possible, this suggests that the classifier must have the recall closer to one, though this would degrade the classifier's ability to precisely classify all observations. This is the result of tradeoff between precision and recall which could not be avoided. The lower precision value would mean that there would be many false alarms, but the higher recall means that we would hardly miss any fire possibility based on weather data. The precision-recall tradeoff is plotted in the above figure and it can be seen that to achieve higher recall value, we would have to compromise with the precision value. We adjusted the threshold value, which is basically a threshold on probability values used by the classifier to determine predicted classes, from its default value of 0.5 down to 0.1, in order to achieve the recall value of 0.9880. This recall value is higher than the 0.9423 recall achieved by Cao et al (2017) for wildfires in Yunnan Province, China. Moreover, the precision value corresponding to our best recall value is precision of 0.2020, which is more than double the precision value of 0.0916 achieved by Cao et al (2017).

Conclusions and Future Directions

Conclusions

Our best classifier was a soft voting classifier involving logistic regression, sgd classifier with loss = "log", and random forest. This was after imputing missing values with the mean, adding polynomial features with a degree of 2, and scaling the data using the standard scaler. However, our best classifier was only able to get a precision of 0.3932 and a recall of 0.7409 on the test set leading to an overall F1 score of 0.5138. When we optimized for recall, we were able to achieve a recall value 0.9880 with a precision value of only 0.2020. This was greater than the recall and precision values found by Cao et al, 2017, but the two datasets are not completely comparable as ours was for California and Nevada, USA, and theirs was for Yunnan Province, China.

The results show that forest fires can't be accurately predicted in the short-term based on current weather conditions. A reason for this is that the weather conditions may not perfectly line up with when a fire occurs (see Future Directions for more) and also our binning had a 1 if a fire occurred in that area in any time of the 10-day window. This means that a fire could have occurred at the very beginning of the window and therefore greatly affected the weather conditions for the rest of the window. However, we can use the results to build a susceptibility map to figure out which areas are most likely to have an ignition. This information can therefore be used to create better exit paths and strategies as well as have the necessary equipment to fight the fires closer.

Future Directions

If we had more time, we would have liked to explore the random train/test split more. The random train/test split got slightly better results from the basic classifiers. So, it might be promising to explore how well we could have done with the random split using ensemble, bagging, and boosting methods. It would have been interesting to see how the best model used when using 2009 and 2010 as the test set fared on a random train/test split of the data. In addition, if we had more time we would have liked to try doing a grid search from KNN on the random test train split. Using the default value for k and p=1 seemed to give pretty decent values, but we did not have the time to run a comprehensive grid search. A simple grid search for KNN using one distance metric and n_neighbors ranging from 1 to 101 going by 2 took over 4 hours alone to run on our dataset. We would have also liked to do grid search for hyperparameters on other classifiers as well. Also, we would have liked to have been able to test out different discretizations of the latitude and longitude features. We only added latitude and longitude in as features at the very end and saw that our results did improve when we added them in as features. As a result, we would want to try and find out the best binning strategy for these features.

As shown, predicting forest fire data over the same time period as meteorological data had a poor F1 score. We predicted whether a forest fire would occur over a 10-day period based on meteorological data over that same time period. In a future project, we would like to use previous meteorological values to predict whether a forest fire will occur at a given time. This would serve the need for predictions a couple days out, e.g., "Based on today's weather in this location, will there be a fire nearby 2 days from now?" This sort of setup would be more useful for real-time evacuations and staging fire crews. This would also allow us to investigate the lag time between foreboding weather and a wildfire ignition.

Advice for Students in DS 5220

The advice we would like to give to future DS 5220 students is to start early on your project. This includes starting early on trying to find teammates and datasets. Also, we suggest working in a group rather than working alone on this project. The reason for this is that not only does having a partner allow you to not have to do the whole project by yourself, but it also allows you to bounce ideas off someone else. The most helpful parts of this project were our group meetings. At these meetings we were able discuss what we had done so far, what had worked, what didn't work, and to bounce ideas off of each other for what would be best to try next. Another piece of advice is therefore that if you do work in a group to make sure you are in constant contact with your group members and try to have a group meeting once a week, even if it is just for 10-15 minutes. You do not want to wait until the last minute to find out that something your partner has been working on hasn't been working, or that they were doing something other than what you thought.

References

Jain et al., 2020. A review of machine learning applications in wildfire science and management. *National Resources Canada Research Press.* Environ. Rev. 28: 478–505

Moorcraft, 2019. These are the top 15 US states with the most wildfire exposure. *Insurance Business America* 25 Oct 2019.

Saha et. al., 2010. The NCEP Climate Forecast System Reanalysis. *Bulletin of the American Meteorological Society* 91.8: 1015-1058.

Short, 2017. Spatial wildfire occurrence data for the United States, 1992-2015 [FPAFOD20170508]. 4th Edition. Fort Collins, CO: *Forest Service Research Data Archive*.

UCAR, 2021. NCEP Climate Forecast System Reanalysis (CFSR) 6-hourly Products, January 1979 to December 2010. Boulder, CO: *UCAR Research Data Archive*. https://rda.ucar.edu/datasets/ds093.0/#!access