Understanding 2014 California Wildfires with Machine Learning

By: Oisin Coveney and Claire Lin

Problem and Motivation

As the Earth continues to warm due to anthropological factors, society must start to adapt to the changing natural environment that accompanies increasing global temperatures and carbon dioxide concentrations. With this increase in heat, ecological systems will experience drier, hotter conditions than those in which they have historically lived. Areas with cooler, wetter climates in the past have experienced temperature increases, and the plant growth associated with these wetter climates have become areas prone to large, disastrous fires that will potentially destroy the lives and habitats of animals, insects, plants, and humans. Recent fires, including the fatal fire in Paradise, California, highlight the growing need for fire prediction services to prevent ecological loss and save lives. Thus, we endeavored to use machine learning techniques to help us determine the main meteorological effects that contribute to forest fires and understand the weather sequences that may indicate the beginning of a fire.

Data Sources

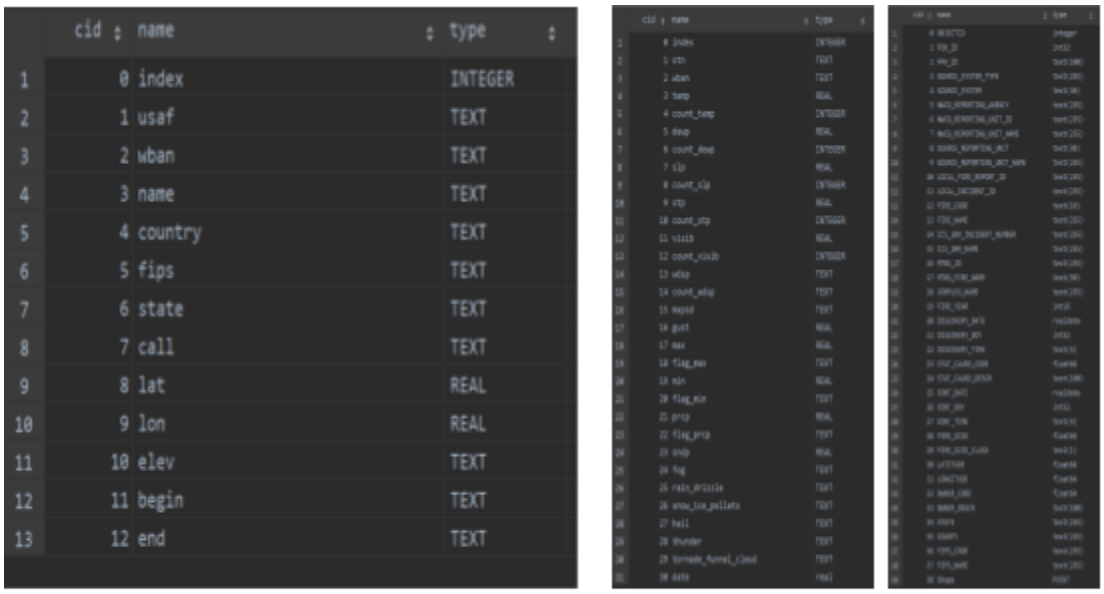
With the help of San Jose State meteorology professor and NASA analyst Rhonda Plofkin, we were able to find an incredibly comprehensive weather dataset from the NOAA (National Oceanic and Atmospheric Administration) for the state of California for 2014. The dataset provides daily weather measurements, including temperature, dew point, atmospheric pressure, and other variables. We also procured a dataset of 1.8 million United States wildfires from 2001 to 2015 from Kaggle, which contained data from over 3000 wildfires in California in 2014 alone. We also obtained California weather station data, which allowed us to cross-reference the NOAA weather station measurements with the fire locations. All of this data was available freely online.

Choosing a single year, 2014, for this endeavor also provided our group with a few practical benefits. We only needed to work with single files from NOAA, who provide

daily weather data as yearly files. With a single year, we were also not lacking any data; for 2014, NOAA provides 4.1 million rows of daily measurements across all of California's weather stations. Also, choosing a single year would allow us to limit the complexity of our code, as we discuss in the next section.

Cleaning the Data

Data processing and cleaning served as the main challenge in this project and required the highest percentage of time that the group devoted to the project. With three different datasets and meteorological terms that our group did not really understand, we endeavored to both understand the data that we had downloaded and manipulate the data in an efficient and comprehensible way. To achieve this task, we used Pandas and Python to programmatically download the data from the NOAA and Kaggle websites, which we then saved to a local SQLite server.



	cid	name	type
1	0	index	INTEGER
2	1	usaf	TEXT
3	2	wban	TEXT
4	3	name	TEXT
5	4	country	TEXT
6	5	fips	TEXT
7	6	state	TEXT
8	7	call	TEXT
9	8	lat	REAL
10	9	lon	REAL
11	10	elev	TEXT
12	11	begin	TEXT
13	12	end	TEXT

The sheer amount of features from the station, fire, and weather data respectively.

Combining the data involved an arduous process that involved many SQL queries and Python code. After downloading and saving all of the data, we first needed to find the closest weather station to each fire in our database. Luckily, the fire dataset offered latitude and longitude points, referring to the center of the fire, which we could cross-reference with the coordinates for each weather station. Using a KDTree, we were able to efficiently match each fire with its closest weather station.

Next, we aimed to label each row in our daily weather table, so we could run supervised machine learning techniques on our weather data. With our new aggregated data set containing weather station and fire data, we joined the tables using weather station ID and (somewhat inefficient) date searching. This combination gave us the data that we needed to begin experimenting.

	cid	name	type	notnull
1	0	level_0	INTEGER	
2	1	index	INTEGER	
3	2	stn	TEXT	
4	3	wban	TEXT	
5	4	temp	REAL	
6	5	count_temp	INTEGER	
7	6	dewp	REAL	
8	7	count_dewp	INTEGER	
9	8	slp	REAL	
10	9	count_slp	INTEGER	
11	10	stp	REAL	
12	11	count_stp	INTEGER	
13	12	visib	REAL	
14	13	count_visib	INTEGER	
15	14	wdsp	TEXT	
16	15	count_wdsp	TEXT	
17	16	mxpsd	TEXT	
18	17	gust	REAL	
19	18	max	REAL	
20	19	flag_max	TEXT	
21	20	min	REAL	
22	21	flag_min	TEXT	
23	22	prcp	REAL	
24	23	flag_prcp	TEXT	
25	24	sndp	REAL	
26	25	fog	TEXT	
27	26	rain_drizzle	TEXT	
28	27	snow_ice_pellets	TEXT	
29	28	hail	TEXT	
30	29	thunder	TEXT	
31	30	tornado_funnel_cloud	TEXT	
32	31	date	TEXT	
33	32	isFire	INTEGER	

The final combination of the weather, station, and fire data. The table contains all of the weather data, while also flagging all days with a fire with the 'isFire' tag.

Methods and Results

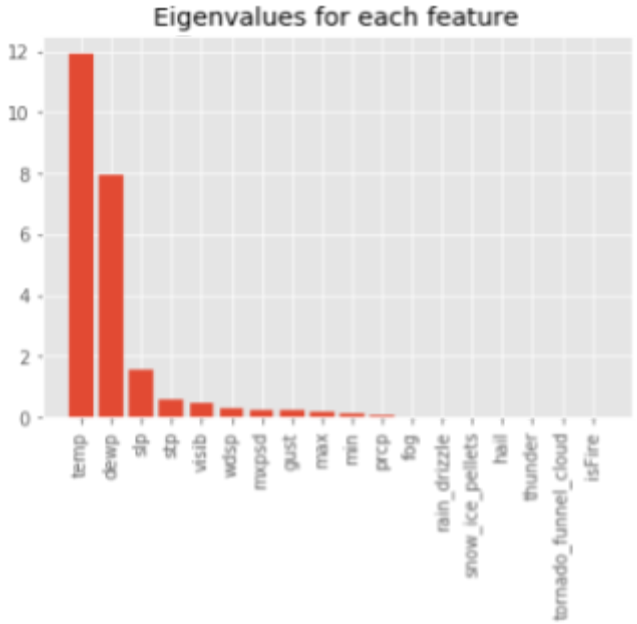
Principal Component Analysis (PCA)

Motivation

After processing the data and looking at the comprehensive set of features that we possessed, we wanted to understand the main contributors to California's forest fires. While we understood that some features, such as snow depth, would probably not heighten the probability of a fire, we also realized that our lack of meteorological knowledge forced us to rely on the statistical methods at our disposal. Thus, we first ran principle component analysis (PCA) on the weather data associated with fires. This technique allowed us to greatly reduce the complexity of our data manipulation and to narrow the number of possible variables from 18 to 3.

Results

With PCA, we determined that the fires were most statistically associated with the temperature and dew point (a measure of air moisture), and to a lesser extent the average pressure in the area (measured in millibars relative to sea level). The chart to the right illustrates the technique's output.



Eigenvalues for each feature after running PCA on the Weather-Fire dataset.

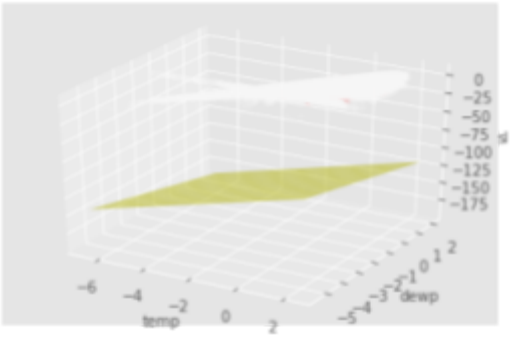
Stacking PCA with a Support Vector Machine (SVM)

Motivation

After greatly reducing the dimensionality of the data, we then wanted to determine if we take advantage of the sheer amount of data that we possessed by attempting to create a fire-prediction support vector machine (SVM). With our reduced data, we were able to create a visualization of the SVM and more easily digest the coefficients and intercepts that our code would output. With this model, we wanted the ability to input any set of weather data and determine whether a fire would be likely or not. By stacking SVM on top of PCA, we hoped to create an ensemble that would provide a more accurate prediction engine.

Results

The SVM was trained on 100,000 weather data points, in which 2% (2000) of the samples were fire-associated weather data. However, despite this amount of data, we found mixed results in our SVM's ability to separate the data points. To remedy this problem, we attempted to re-process the data and ensure that each row was classified correctly. This problem may be attributed to our group's lack of knowledge in Python graphics tools.

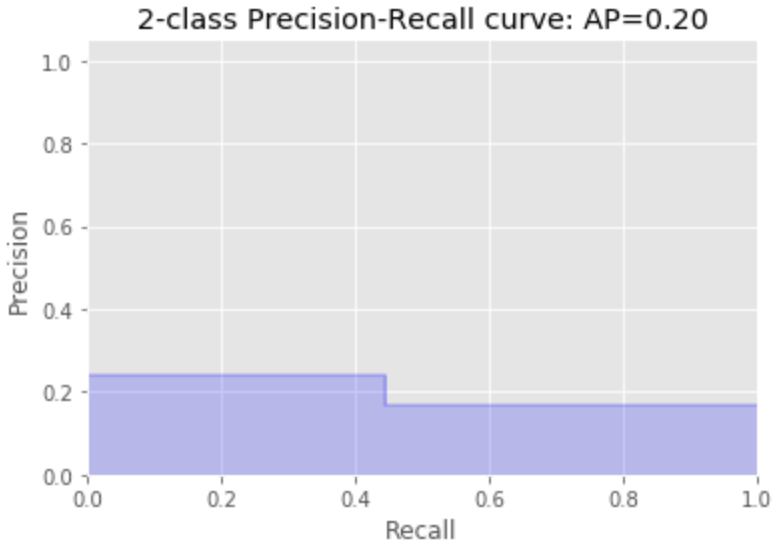


The SVM, trained on a sample of 100,000 weather data points. The hyperplane is highlighted in yellow, and fire data is highlighted in red.

However, we found much more promising results in our accuracy and precision-recall curves. We achieved relatively scores in our ability to classify fires based on the given weather data. Sampling 5000 non-fire and 1000 fire weather data points, we achieved approximately 68% accuracy, 87% precision, and 72% recall. The latter two scores contributed to an extremely low value for the area under the Precision-Recall curve, achieving a value of 0.20. Outputs of these values are provided below.

	precision	recall	f1-score	support
0	0.87	0.72	0.79	5000
1	0.24	0.45	0.31	1000
accuracy			0.68	6000
macro avg	0.55	0.58	0.55	6000
weighted avg	0.76	0.68	0.71	6000

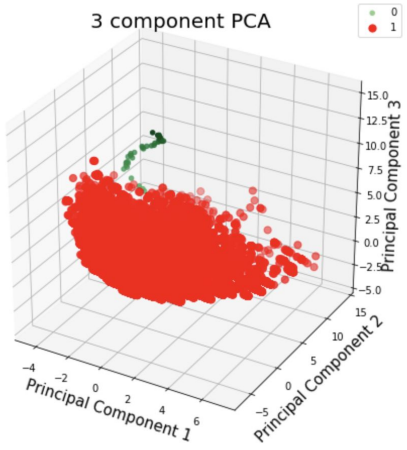
We achieved higher-than-expected scores in our accuracy, precision, and recall values.



LDA (Linear discriminant analysis)

Motivation

After seeing that the training on SVM with PCA achieves only 68% accuracy, we hope to improve the results. By plotting the data points in PCA, it is shown that the means of fire data (red dots) and no-fire data (green dots) are separated and there is little overlap between the scatter of the two dotted groups, which is ideal for using LDA. As the result, we proposed using LDA as another approach for data dimension reduction.



Results

With LDA, the final dimensionality of the data can only be reduced to one dimension due to the data having only two classes: fire, no-fire. The table below shows the reduced dimension of the data.

	lda component
0	0.670478
1	-1.371285
2	-1.348317
3	-1.365512
4	0.640190
...	...
4115577	-1.414229
4115578	0.600539
4115579	0.696619
4115580	0.686915
4115581	0.684346

4115582 rows × 1 columns

Stacking Neural Networks with a LDA

Motivation

After greatly reducing the dimensionality of the data using LDA , we then wanted to create a fire-prediction sequential neural network model to compare with the SVM-PCA model previously developed.

Results

We sampled all the data points in the database, about 4000000 pieces. We built a three-layer architecture with ReLU and sigmoid as activation functions. With the reduced data dimension and setting the batch size for each epoch to 10000, the time of training 100 epoches of neural networks has reduced to no more than 3 minutes. The accuracy of the neural network model reaches 99% . By stacking neural network on top of LDA, we are able to create a model that would provide a more accurate fire prediction engine given a piece of weather data.

```
classifier.evaluate(X_test, y_test)
```

```
1234675/1234675 [==============================] - 17s 14us/step
[0.004938211188290818, 0.9994152188301086]
```

Accuracy (on the left column of matrix) reaches 99%.

Note that the accuracy can be affected by the unbalanced number of fire data and non-fire data

Next Steps

In the future, we would want to pursue greater breadth and depth in exploring the data. Ideally, we would sample more years from the NOAA website and create an automated pipeline that would allow us to associate fire, weather station, and daily weather data quickly and efficiently. We also would want to explore HMM, neural networks with memories, stacking, and boosting to more effectively determine the best features to use in predicting fires. More data would also allow us to make time-series models, which we could then test against climate prediction models to predict the locations and effects of fires in the future.

References

Datasets:

NOAA Climate Data:

- *https://www.ncdc.noaa.gov/cdo-web/datatools/lcd*
- *Alternate (when NOAA web servers were down):* *https://www.kaggle.com/noaa/gsod*

Wildfire Dataset:

- https://www.kaggle.com/rtatman/188-million-us-wildfires

Station Data:

- https://bigquery.cloud.google.com/table/fh-bigquery:weather_gsod.stations

The Best Reference:

Stamp, Mark. *Introduction to machine learning with applications in information security*. Boca Raton, FL: CRC Press, Taylor & Francis Group, 2018. Print.

Sk-learn guides:

- https://medium.com/datadriveninvestor/building-neural-network-using-keras-for-classification-3a3656c726c1
- https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60
- https://medium.com/@sebastiannorena/3d-plotting-in-python-b0dc1c2e5e38