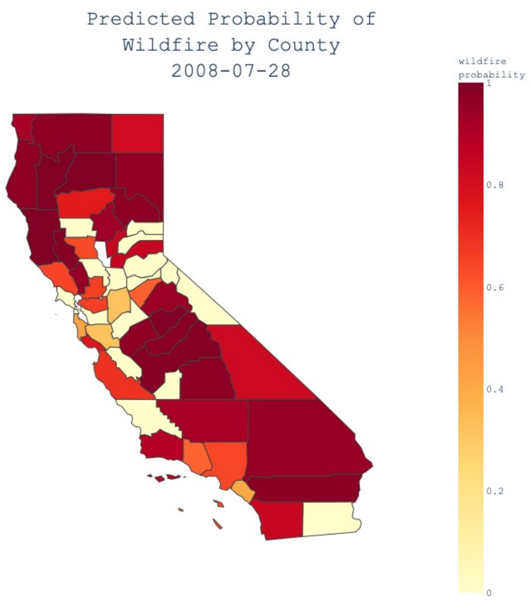
Random Forest Fires:
Modeling Wildfire Probability with Weather

Matt Elser
elserm@gmail.com

September 9, 2020



1 Requirements

Running the code requires the following modules:

- `pandas`
- `numpy`
- `sklearn`
- `requests` - for accessing FCC area information API
- `plotly` - for creating graphs of geographic areas
- `kaleido` - required for plotly to write graphs to images

As well as the code and data available at https://github.com/mattelser/randomForestFires. The tool `ffmpeg` was used to combine graphs into an animation, viewable at the aforementioned github link. All work was done on a unix system and no validation was feasible on windows. Those wishing to regenerate all baked data can do so by first running `bakeData.py` to process all data in `//cleanedData/`, generating `//bakedData/fireWeatherData.csv`. Running `runModel.py` processes this baked data and generates `//bakedData/predictedFireData.csv`, which contains predicted probabilities of having a fire in each county on each day in our data window. To generate graphs for all these predicitons, run `plotCounties.py`.

Introduction

California wildfires occur yearly, and can quickly become catastrophic. Being able to predict when and where a wildfire may start can be extremely useful, and can save lives, homes, and livelihoods. I was interested in creating a model that could be the beginnings of such a useful tool. I chose to use a random forest model because it generally avoids overfitting, handles highly correlated features (such as cold and rain in the Bay Area) well, and I did not believe the relationships involved would necessarily be linear. Intuitively, the problem seemed like it lent itself well to a decision tree, which would mimic the way I would personally asses fire risk: It's hot today, has it been hot recently? It has, but has it rained recently? It has not, so the fire risk is likely high. The model was trained on weather data from 2008 through 2013, pairing it with all recorded "large fires" that burned at least 300 acres recorded during that time.

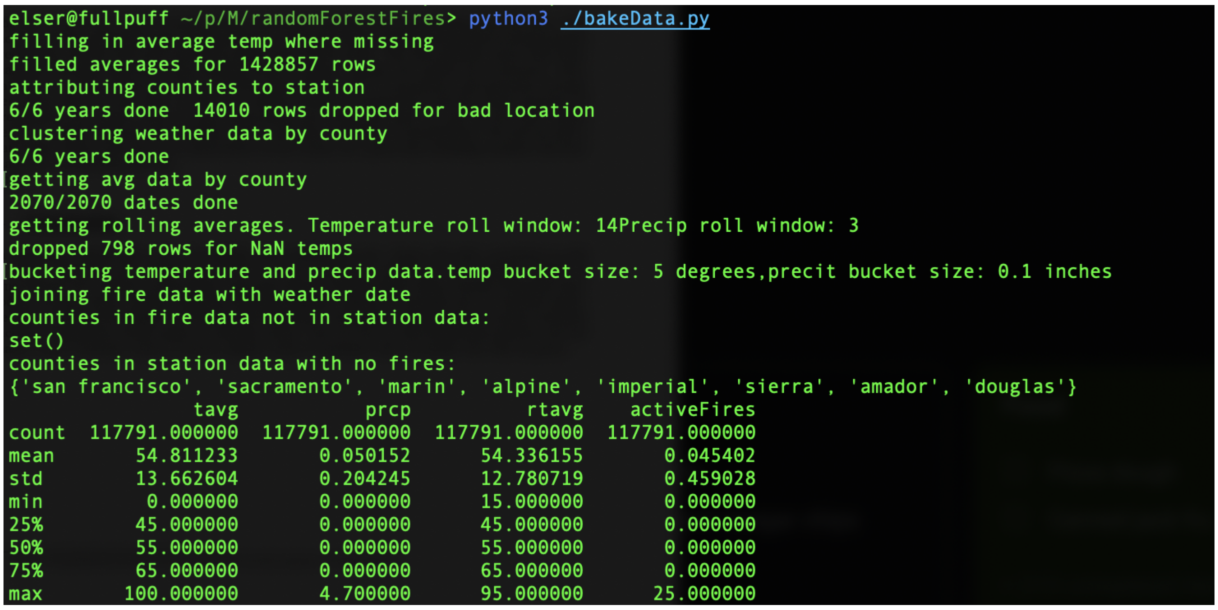
2 Data Gathering

Weather data is readily available from the National Climatic Data Center (NCDC) and the National Oceanic and Atmospheric Administration (NOAA), and wildfire data is made available by the California Department of Forestry and Fire Protection (CalFire), however obtaining sufficient data in a usable form still proved challenging. Raw data from NOAA [1] (available in `/rawData/weather`) had to be obtained through multiple manually-clicked requests, as no API existed to query and a limit on request size meant several requests had to be made per year of data. CalFire makes their data available via "Redbook" pdfs [2] , from which the contents of tables was manually extracted using a tool called `Tabula`. The fire data is only by county, while the weather data was by station, which had no county attributed. Separately, a collection of latitudes and longitudes was obtained (again from NCDC/NOAA) for each station. This allowed for obtaining county via an API made available by the Federal Communications Comission (FCC) [3] , which accepts a latitude/longitude pair and returns geographic information. Included in this information is both county name and Federal Information Processing Standards (FIPS) code, which was one key part in graphing information by county. The second key part to graphing by county is pairing each FIPS code with a polygonal outline of that county (defined by points of latitude/longitude). This was available via an example in the documentation for the graphing module `plotly` [4] in the form of a "geojson" file defining the shape of all counties in the US. A paired-down version of this geojson file containing only California is available in `//CalCountyGeoJson.json`.

3 Code Description

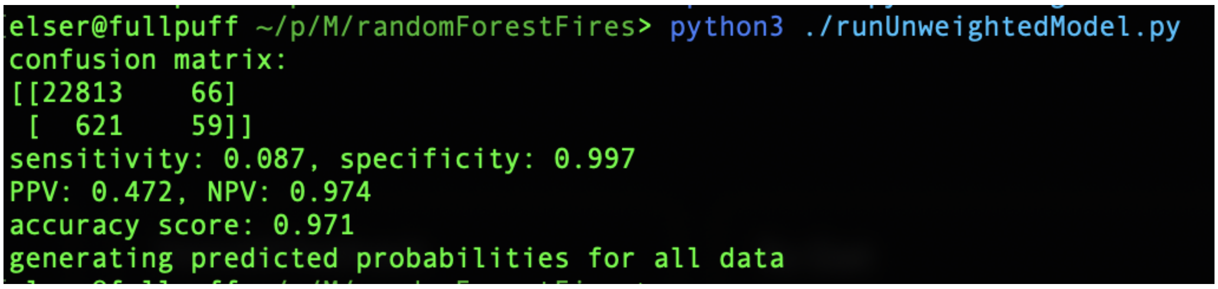
The bulk of the code centers around joining, clustering, cleaning, bucketing, and extrapolating from the data. This includes all data gathering steps mentioned above, other than those explicitly mentioned as being manual (requesting weather data and extracting tables from CalFire pdfs). The data pipeline starts with these manually extracted data files, then develops them in stages, "baking" the results out in between. First, `bakeData.py` takes all the per-station-per-date observations from `\cleanedData` and generates average per-county data for each date, as well as a rolling average of temperature and the recent presence of precipitation per county. Measurements are also bucketed for better usability in a random forest model. Once we have observations per county per date, we join the fire data, which begins as various fields per-fire (notably, county, start date, and date of containment). We end up with a data frame where each row is an observation of one county on a given day, including weather and how many fires are burning that day. A description of each action is printed to stdout:

[1] https://www.ncdc.noaa.gov/cdo-web/search
[2] https://www.fire.ca.gov/stats-events/
[3] https://geo.fcc.gov/api/census/#!/area/get_area
[4] https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json



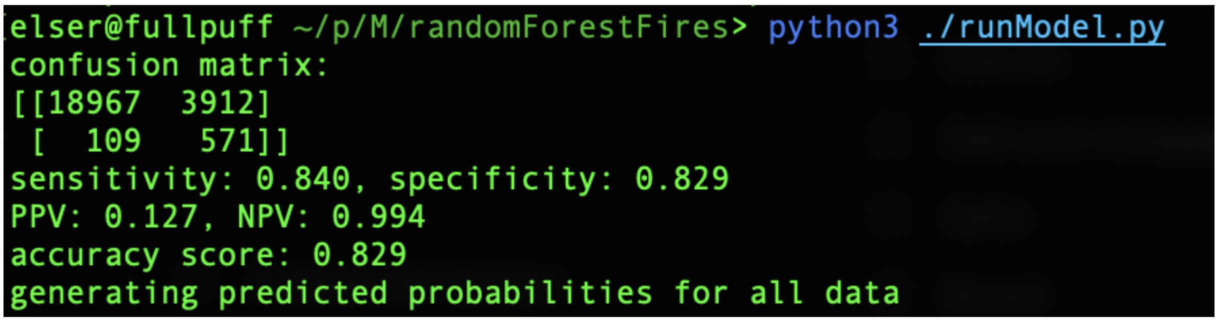
```
elser@fullpuff ~/p/M/randomForestFires> python3 ./bakeData.py
filling in average temp where missing
filled averages for 1428857 rows
attributing counties to station
6/6 years done  14010 rows dropped for bad location
clustering weather data by county
6/6 years done
getting avg data by county
2070/2070 dates done
getting rolling averages. Temperature roll window: 14Precip roll window: 3
dropped 798 rows for NaN temps
bucketing temperature and precip data.temp bucket size: 5 degrees,precit bucket size: 0.1 inches
joining fire data with weather date
counties in fire data not in station data:
set()
counties in station data with no fires:
{'san francisco', 'sacramento', 'marin', 'alpine', 'imperial', 'sierra', 'amador', 'douglas'}
                tavg           prcp          rtavg    activeFires
count  117791.000000  117791.000000  117791.000000  117791.000000
mean       54.811233       0.050152      54.336155       0.045402
std        13.662604       0.204245      12.780719       0.459028
min         0.000000       0.000000      15.000000       0.000000
25%        45.000000       0.000000      45.000000       0.000000
50%        55.000000       0.000000      55.000000       0.000000
75%        65.000000       0.000000      65.000000       0.000000
max       100.000000       4.700000      95.000000      25.000000
```

This is the data that feeds the model. A `RandomForestClassifier` is fed per county per day observations of average temperature, precipitation level, a rolling average of temperature, whether there has recently been precipitation, and what county these observations were in. Notably, date is not given to the model, even though these observations are per-date, because dates do not contribute to fires and should therefore not be part of the decision trees generated by our random forest model. The target is whether there is a fire in that county. Initial runs of the model were very good at predicting when/where there would not be a fire, and mediocre at predicting when/where a fire would occur. Fires are relatively rare in the dataset, so it makes sense that they would not be predicted often, however in this case false negatives could be catastrophic and therefore reweighting of the model was necessary. Observations with fires were given a higher weight, and that weight was further scaled by the number of fires in the county. Here are the results of the model before weighting:



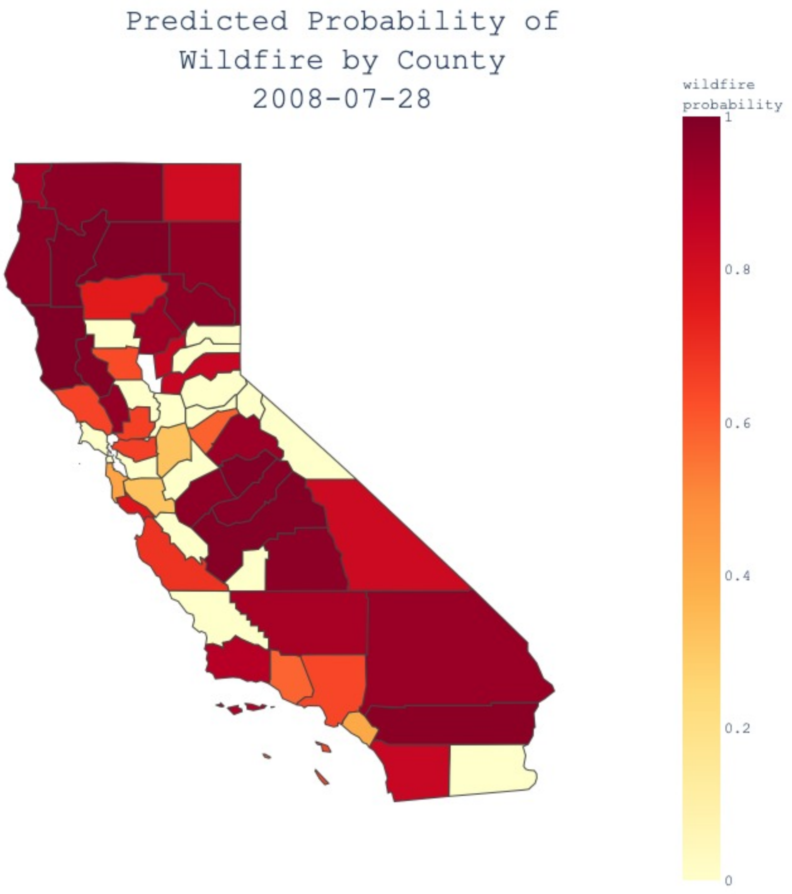
```
elser@fullpuff ~/p/M/randomForestFires> python3 ./runUnweightedModel.py
confusion matrix:
[[22813    66]
 [  621    59]]
sensitivity: 0.087, specificity: 0.997
PPV: 0.472, NPV: 0.974
accuracy score: 0.971
generating predicted probabilities for all data
```

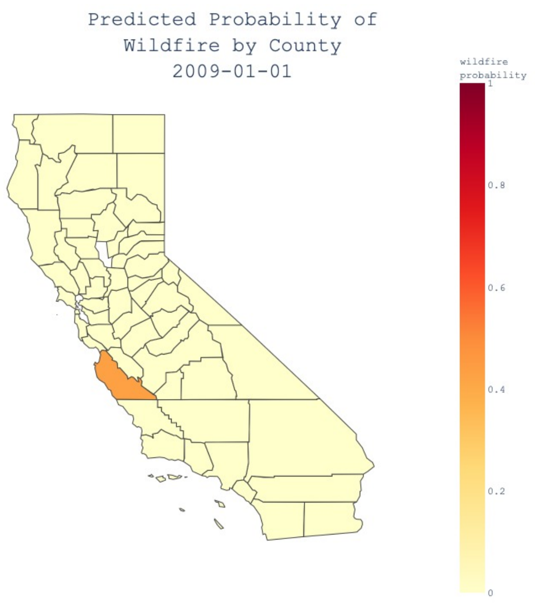
And after reweighting, the sensitivity improves nearly tenfold. Our positive predictive value decreases, however the cost of a false positive is extremely low in comparison to a false negative, so our gains seem more than worth it.

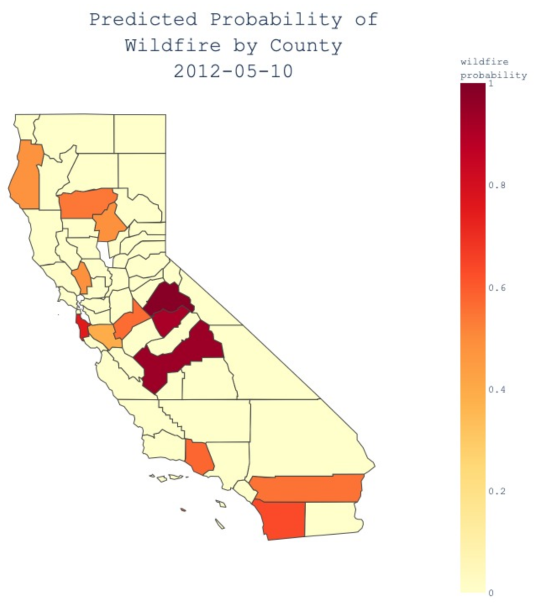


```
elser@fullpuff ~/p/M/randomForestFires> python3 ./runModel.py
confusion matrix:
[[18967  3912]
 [  109   571]]
sensitivity: 0.840, specificity: 0.829
PPV: 0.127, NPV: 0.994
accuracy score: 0.829
generating predicted probabilities for all data
```

Using this model, we can generate a predicted probability of having a fire in each county for each day, given the weather data. These predicted probabilities were generated for all per-county weather observations, allowing the creation of graphs like these:







Such graphs were created using `plotly`, which can generate choropleths using the generated data and geojson described above. Each observation was given a probability and graphed, which, when combined using `ffmpeg`, create an animation showing predicted probability of fire by county over time. This animation can be viewed at the github repository mentioned in the Requirements section, or can be generated by following the steps listed in the Requirements section then `convertFramesToGraph.sh` .

4 Conclusion

With the improved weighting, the model seems to be decent at predicting when/where a fire may occur. Given the sensitivity of 0.84, the model successfully predicted 84% of fire times and locations outside the training data set. With a positive predictive value of approximately 0.13, it would seem that a fire occurs only 13% of the time the model states a high probability, but again given the high consequence of a wildfire and the low consequence of a false positive, this seems acceptable. This model is assuredly less sophisticated than those used by organizations performing actual fire prevention and containment, and given the size of some counties the model is not particularly precise geographically, but given relatively few inputs it is able to predict fire times and locations decently accurately.