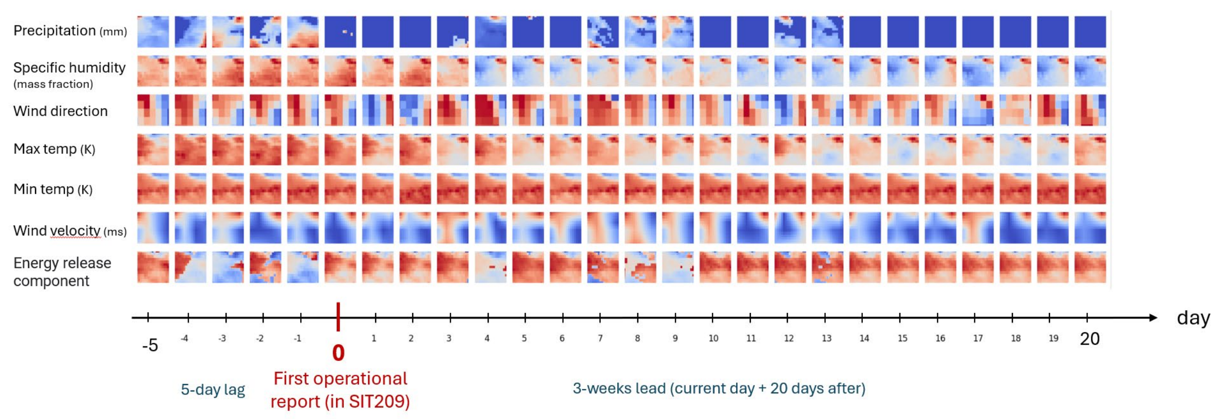
Wildfire spread prediction

Satellite Image Extraction

For all 5k fires between 2015 and 2018, I have selected and downloaded all the corresponding satellite images for 26 days across 7 features: precipitation, median temperature, max temperature, min temperature, wind speed, energy release, specific humidity.

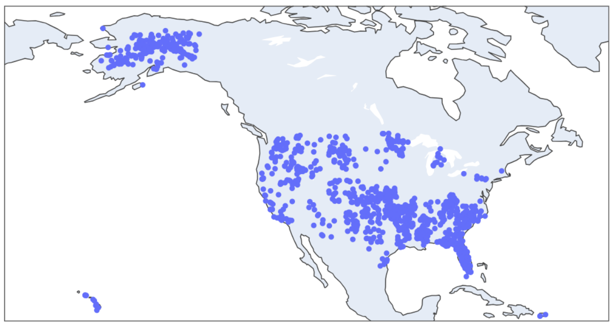


The major problem I faced was that I was not trying to get any fire, but the specific images for a certain fire given his location and time. This has induced some changes in how I had to store the images so that errors would not mess everything up (for instance, if a fire fails to be downloaded, the others are downloaded and matched with the right fire).

Data preprocessing

Then, I had to build a multimodal Dataset: for each report, get the associated image using the image tensor of the fire, selected on the day corresponding to the report.

I have dropped 3k reports (corresponding to 1k fires) whose satellite images could not be found. I first suspected those were fires outside CONUS, but apparently this was not the case:



The pytorch dataset class that can iterate through the reports, outputting for each report a tuple (features, images, target).

Features: tabular array of size 49, preprocessed from the decoded SIT209 database

- Behavior: describes the behavior of the fire, such as s. In SIT209, agencies can input up to 3 behaviors. Here I just selected the first one, and one-hot encoded it, since there is no natural order. It represents 16 features.
- Fuel: fuel model that best represents the primary carrier of the fire. Same, one hot encoded here the main fuel. Represents 13 features.
- Cause: whether caused by human, lightning, other, unknown
- Fire behavior: a general characteristic of how active the fire is (minimal, moderate, extreme)
- Method: the method used to control the fire
- Area and area diff
- Injuries, fatalities
- Total personnel, total quantity of supply items

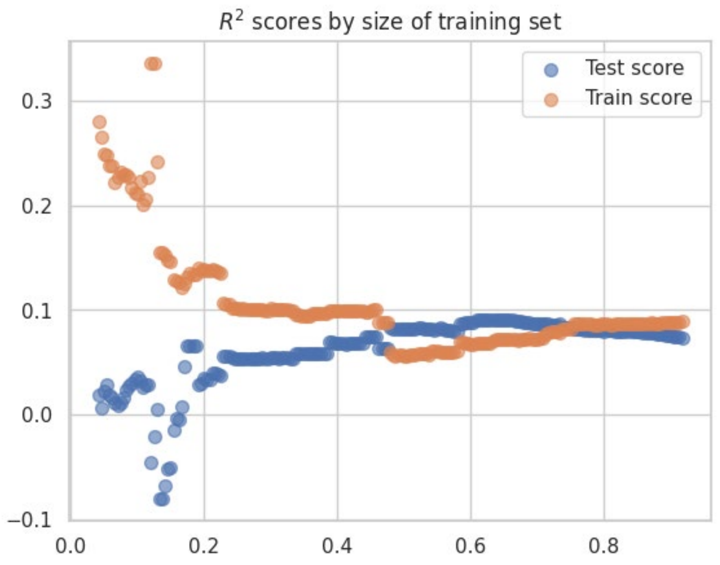
Images: from Google Earth Engine

Target: the target is next-report area difference with the previous report Aj+1 – Aj. This is subject to further preprocessing.

Once this dataset is built, we can setup run a simple linear regression on the features.

Train/test splitting

To split the data, we want to operate a chronological split to prevent leaks in time (using the future to inform the past). This was easy to obtain since my dataset is ordered chronologically. To analyze how much signal there is to learn from the samples, I swept the size of the training set. For each size of training set (cut in chronological order) I ran a ridge linear regression. Here is the visualization of the R^2 scores.

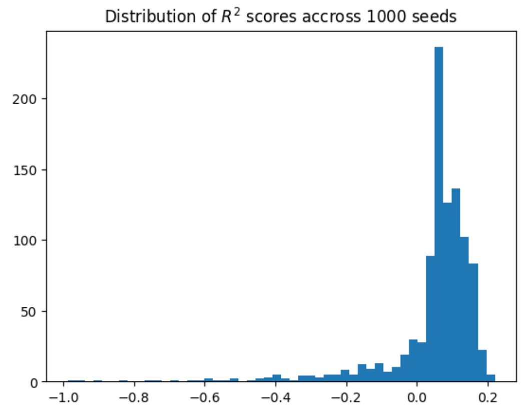


Observations

1) maximal R^2 score on test set is 0.091
2) This looks almost like a loss vs epoch curve that we could observe when training deep learning models. What is remarkable here is the quick convergence towards near-optimality: if the train set is 20% of the dataset, the R^2 score on the test set is 0.05.
3) The curves are not perfectly monotonic. This might be because my cuts are not in respect to the fires but by reports: a same fire could have reports in both train and test set.

On a side note, I am not sure people usually study these types of curves, but I feel like they are quite informative: showing how much data you should gather to predict the future. This gives a sense of the stationarity of the data generation process. I am surprised it is so stable given the noise and biases arising from the human input dataset.

Out of curiosity I wanted to check what happens if we don't restrict the order to be chronological and just randomly splits reports (therefore, a fire can have reports both in train and in test sets). We get instable results. That means that the **look-ahead bias matters** here.

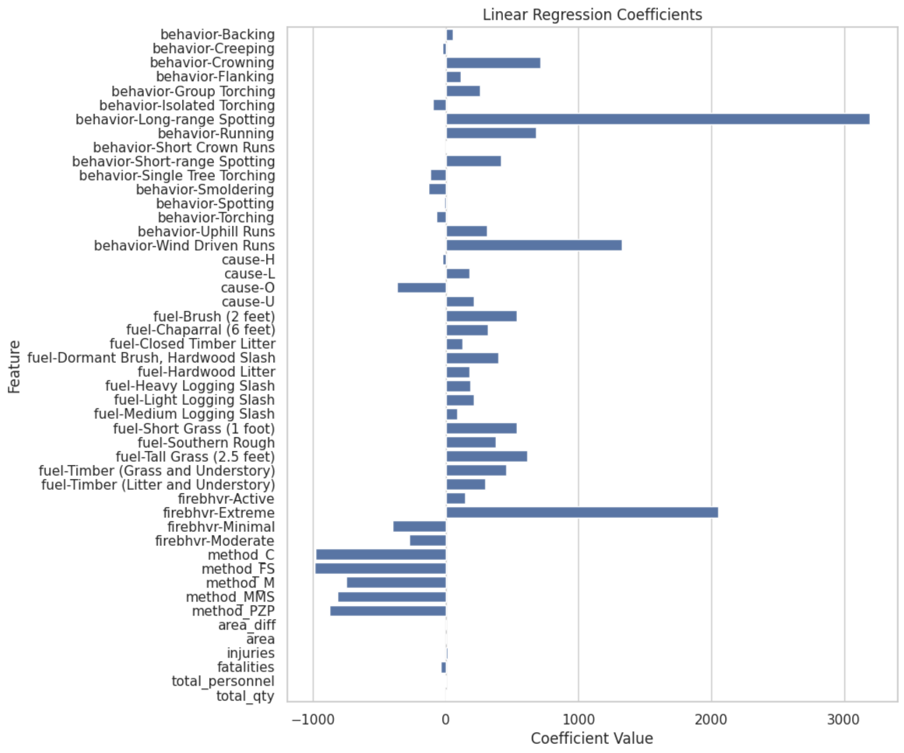


I selected an index that would split train-test at 79.2%, and respect the fire integrity (no fire could have a report in both splits)

Regression analysis

I ran regressions using the common models. No significant differences have been found between normal, lasso, and ridge regressions.

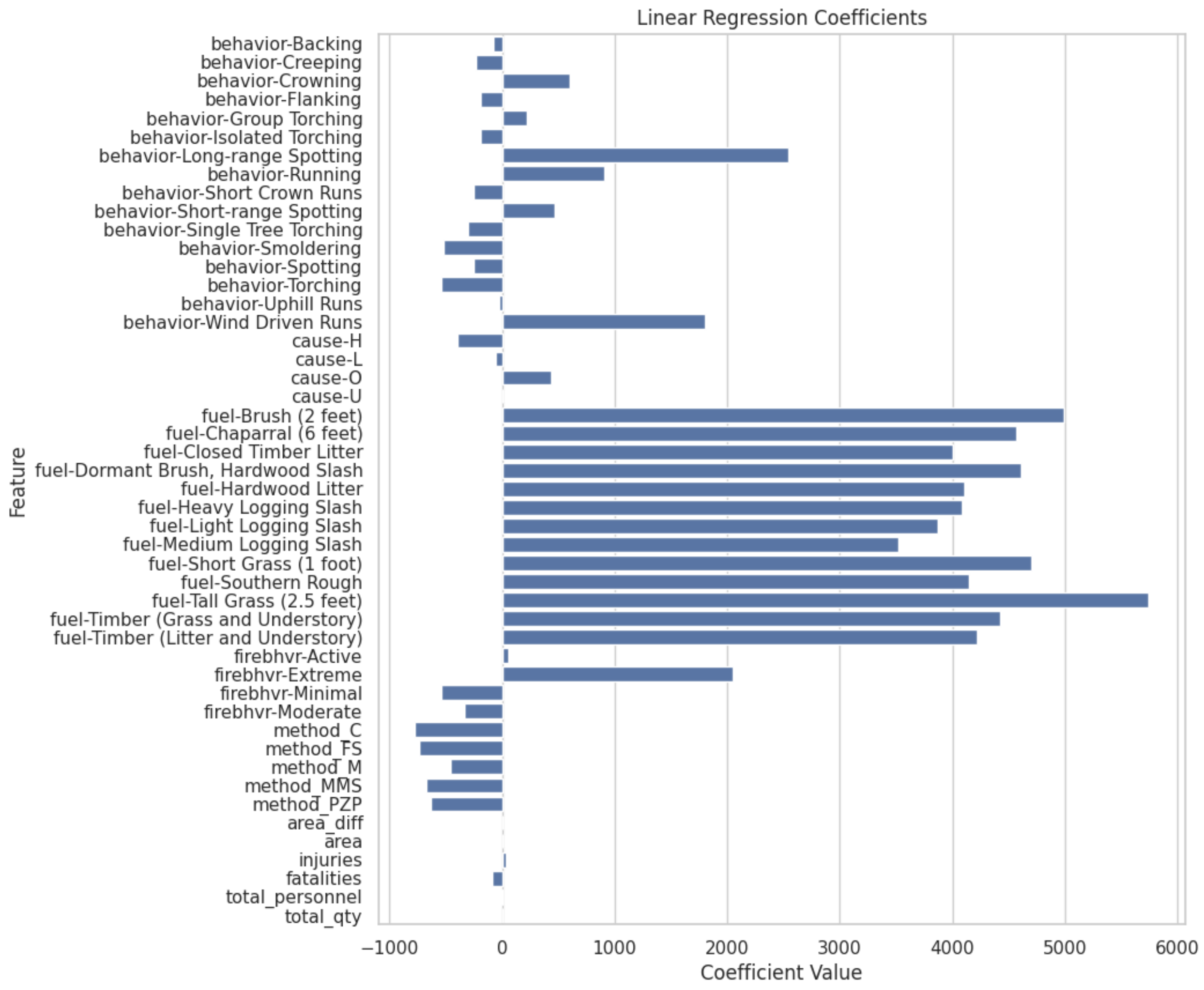
Using ridge regression, we get the following regression coefficients:



Observations

1) Largest coefficients: behavior = long-range spotting, fire behavior = extreme, behavior = wind drive runs. This makes sense since those features are correlated with big fires.
2) Surprisingly, other features we would expect to be correlated with the target have a null coefficient: area, previous area difference. While the coefficient don't actually indicated correlation because of multicollinearity, this is unexpected.
3) Some features all have the same sign coefficient. For instance, all methods have a negative coefficient. I should check if all reports have a method, otherwise I don't understand.

Since the target is (too) often equal to zero, we might want to look at the quality of the signal on the subset of the data where the target is not 0. This leaves us with 5k reports. Running a ridge regression, using 80% of train.



Experimentation pipeline

I built a pipeline to experiment with these variables:

- Models: naïve model, linear regression, ridge, lasso, …
- Metrics: MSE, RMSE, MAE, Median AE,

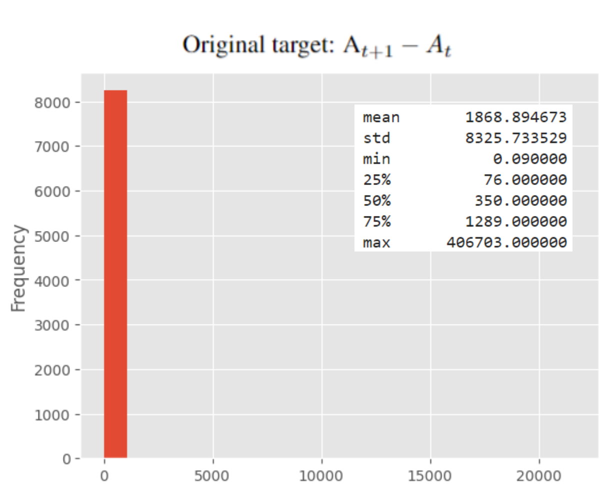
The biggest complexity here is not the design nor the implementation of the experiments, but creating debugging features (searching for fires, searching for features, selecting subsets before vs after analysis)

Target analysis

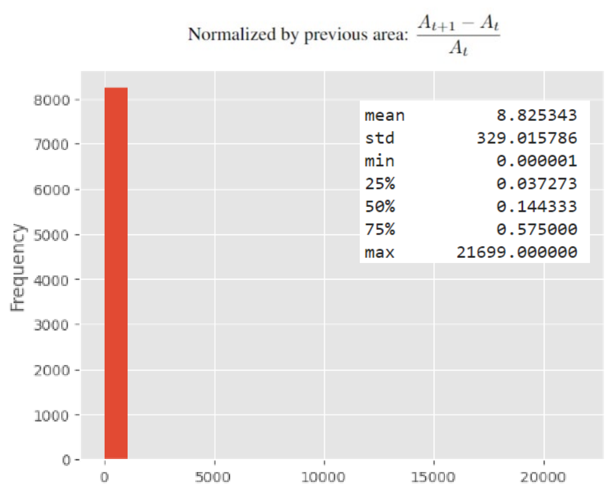
I have run the whole pipeline with the whole dataset. Since 15735 samples (= 65% of the dataset) have a target next_area_diff equal to zero, the models would basically learn to predict zero.

Therefore, I led a study on the remaining part of the dataset, and computed different target transformations in order to get a “natural” distribution.

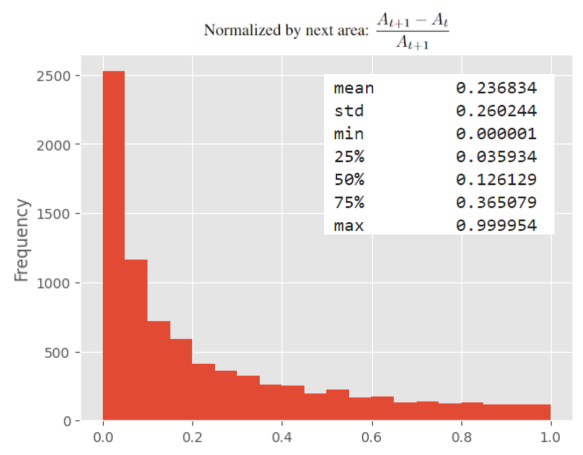
Distribution plots of the target (area difference between today and next report):



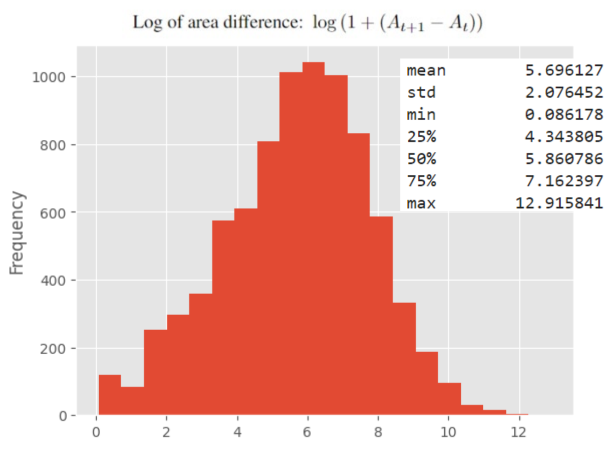
```
mean       1868.894673
std        8325.733529
min           0.090000
25%          76.000000
50%         350.000000
75%        1289.000000
max      406703.000000
```



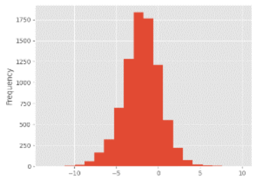
```
mean         8.825343
std        329.015786
min          0.000001
25%          0.037273
50%          0.144333
75%          0.575000
max      21699.000000
```



```
mean     0.236834
std      0.260244
min      0.000001
25%      0.035934
50%      0.126129
75%      0.365079
max      0.999954
```



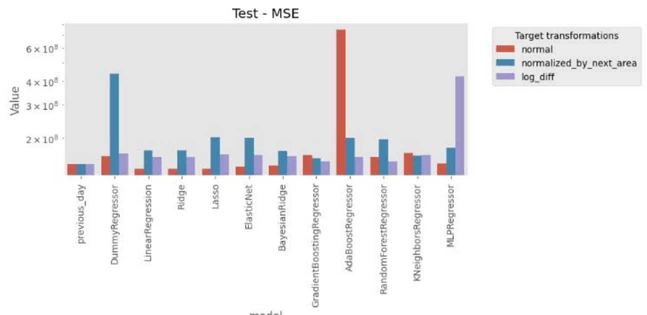
```
mean      5.696127
std       2.076452
min       0.086178
25%       4.343805
50%       5.860786
75%       7.162397
max      12.915841
```

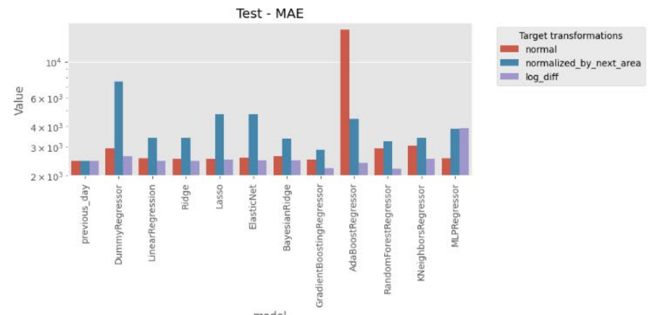
Attention! t+1 is not tomorrow, it is just the next report (even though it often is the same thing). It is also possible to take log of the area difference normalized by an area. For instance, when taking the log over area difference normalized with area previous day: 

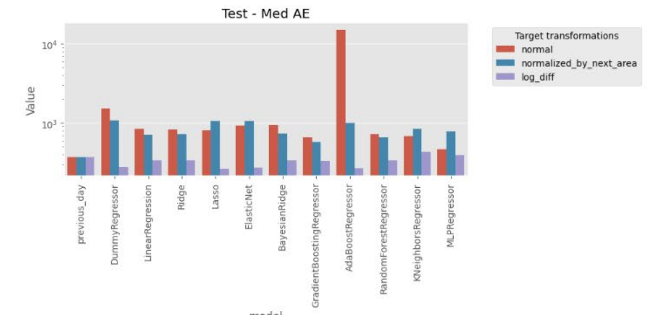
.

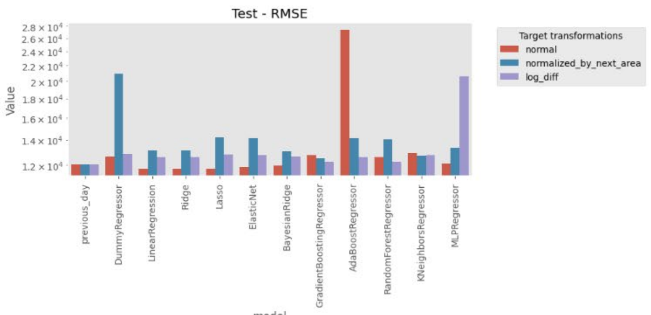
Pipeline output:

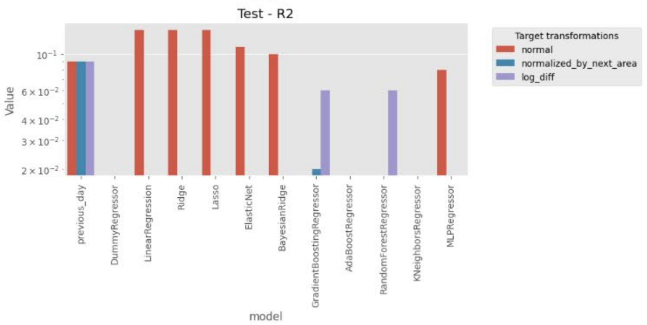
For better readability, I removed some models that were either overfitting (decisionTreeRegressor) or underfitting (some models "robust to outliers" such as HuberRegressor).











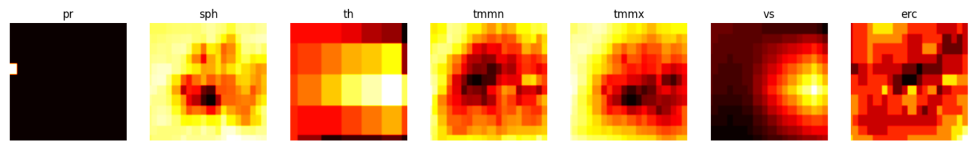
Conclusion: R^2 scores are hardly over 0.12. The learnability is small because the target is very noisy. When the human fills the ICS209 survey, he can be pretty confident about the number of people sent, the behaviour of the fire, the strategy deployed, but he has hard time to assess the size of the fire + he does not have an incentive to fill it accurately.

Adding satellite images

I ran CV models with various freezing depth, various subsets => no learning.

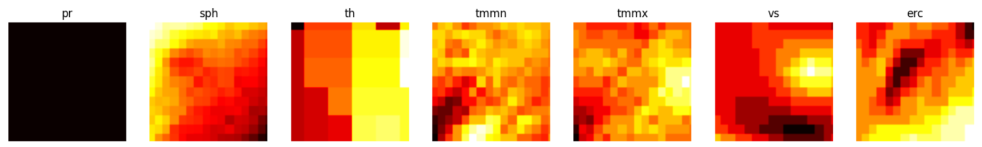
- Target is bad + deep learning not super good for regression anyways

```
Report index: 16416, fire id: 2017_7362311_EAST FORK
```



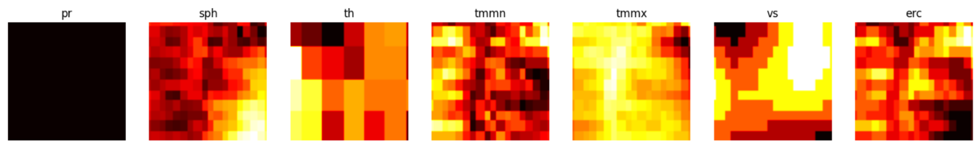
```
Features                                        Diff area
behavior-Short Crown Runs
cause-U
fuel-Timber (Grass and Understory)
firebhvr-Moderate
method_FS
area_diff: 1103.0
area: 21103.0                                   0.0
injuries: 0.0
fatalities: 0.0
total_personnel: 100.0
total_qty: 26.0
day_since_first_report: 4.0
```

```
Report index: 22672, fire id: 2018_9214372_SURPRISE
```



```
Features                                        Diff area
behavior-Wind Driven Runs
cause-U
fuel-Tall Grass (2.5 feet)
firebhvr-Active
method_FS
area_diff: 600.0
area: 600.0                                     1400.0
injuries: 0.0
fatalities: 0.0
total_personnel: 28.0
total_qty: 13.0
day_since_first_report: 0.0
```

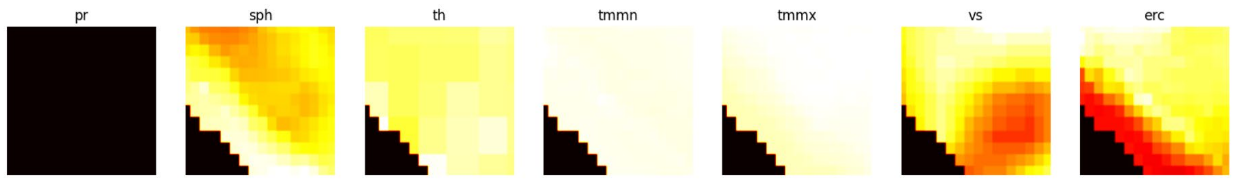
```
Report index: 13960, fire id: 2017_7269467_LODGEPOLE COMPLEX
```



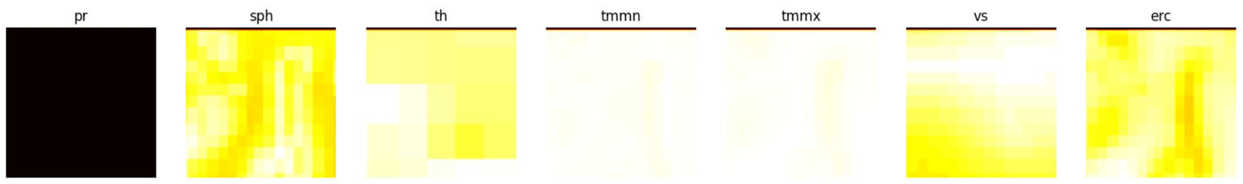
```
Features                                        Diff area

cause-U
fuel-Timber (Grass and Understory)
firebhvr-Minimal
method_FS
area_diff: 0.0
area: 270743.0                                  0.0
injuries: 0.0
fatalities: 0.0
total_personnel: 30.0
total_qty: 5.0
day_since_first_report: 19.0
```

Also, while many errors have been corrected, the data extracted remains unsatisfactory. Three sources of problems:

1. Missing pixels at the geographic boundaries



2. Border effects: sometimes one or two rows are missing. Tried a few things that seemed to have improved the quality but have not figured out why exactly.



3. Size of pixels: for now there is no interpolation, so the images seem to be patches of different shapes.