BASIC INFO II

Name: Andrew Wheeler

Hometown: Raleigh, NC

Pic:

url: https://andrewpwheeler.com/

MODEL DOCUMENTATION III

1. *Who are you?*

My name is Andy Wheeler. I have a background in predictive modelling in social science applications (PhD is in criminal justice). Currently I am a data scientist in the healthcare field working on predictive models relating to aspects of insurance claims.

2. *What motivated you to compete?*

I have been following DataDriven for a bit, decided I had enough time when this competition came out and that I might be able to improve upon the baseline notebook. Early results were very promising, so continued to improve upon that baseline over the whole competition.

3. *High Level Summary of Approach*

I generate tabular data based on date, location, and elevation of the sample. I additionally used k-means segmentation to try to identify the lake area in the Sentinel imagery, and then extract out the RGB values for that segmented lake area.

There was a significant amount of spatial/temporal autocorrelation. My modelling approach identified spatial clusters based on the holdout set, and those ended up providing a pretty decent increase in accuracy.

4. Useful tables/charts/graphs

In the root of my public repo, you can check out some of my notes and results, https://github.com/apwheele/algaebloom/blob/main/model_strategy.ipynb , that has some background, stats, validation strategy, etc.

5. *Copy and paste 3 most impactful parts of the code*

```
# identifying clusters
# Looking at train/test
# there are a few clusters, want
# to make sure to predict these well
def cluster(x):
    lat, lon = x[0], x[1]
    if (lat < 41) & (lon < -116):
        # cali
        return 7
    elif (lat < 41) & (lat > 36.29) & (lon < -92.9) & (lon > -102.2):
        # midwest
        return 6
    elif (lat < 38.14) & (lat > 33.26) & (lon < -74.8) & (lon > -85.52):
        # carolina
        return 2
    elif (lat < 43) & (lat > 38.7) & (lon < -75.4) & (lon > -83.55):
        # erie
        return 3
    elif (lat < 43.1) & (lat > 40.7) & (lon < -69.5) & (lon > -74.6):
        # mass
        return 4
    elif (lat < 49.6) & (lat > 41.5) & (lon < -83.55) & (lon > -104.56):
        # dakota
        return 1
    else:
        # other
        return 5

# this is the code to identify the clusters
# adding this simple ordinal factor into the models increased accuracy by
# quite a bit
```

```
# I ensemble different boosted models together in the end
# If you pass in multiple models
# this will ensemble them, presumes regressor models
class EnsMod():
    def __init__(self, mods, av_func = 'mean'):
        self.mods = mods #should be dict
        self.av_func = av_func
    def fit(self,X,weight=True,cat=True):
        for key,mod in self.mods.items():
            mod.fit(X,weight=weight,cat=cat)
    def predict(self,X):
        res = []
        for key,mod in self.mods.items():
            res.append(mod.predict(X))
        res_df = pd.concat(res,axis=1)
        if self.av_func == 'mean':
            pred = res_df.mean(axis=1)
        return pred
    def predict_int(self,X):
        pred = self.predict(X)
        pred = pred.clip(1,5).round().astype(int)
        return pred
```

```
# I identified that even day of the week
# (suggesting human bias in when samples were taken)
# ending up being a factor predicting algae blooms

def dummy_stats(values,begin_date):
    vdate = pd.to_datetime(values,errors='ignore')
    year = vdate.dt.year
    month = vdate.dt.month
    week_day = vdate.dt.dayofweek
    diff_days = (vdate - begin_date).dt.days
    # if binary, turn week/month into dummy variables
    return diff_days, week_day, month, year
```

6. *Machine Specs*

Only used CPU, the biggest hurdle is in downloading the sentinel data (even on CPU the k-means segmentation does not take very long for a single image, but can get rate limited, so I used planetary compute resource in the end)

CPU: Intel i3-9100F CPU @ 3.6 GHz

RAM: 32 gig (I don't think was really necessary though)

OS: Windows

Train duration: less than 20 second even for all 3 ensembled models.

Inference duration: very fast (under 2 seconds) if the data is already downloaded. If grabbing the sentinel and DEM data on the fly, it can take several seconds to download the data and pre-process the data to input into the model.

7. *Anything we should watch for?*

I made a mistake early in the competition, and used landsat-7 data (instead of landsat-8 or 9). It is possible that incorporating this data would result in better predictions. (Current solution sets landsat data samples to missing data.)

8. *Did you use any tools not in submission?*

Only thing I used was a planetary computer resource hub. This was to not get rate limited downloading sentinel data though, not for the actual compute.

9. *What features did you select in your model and why?*

So I spent a lot of time trying to get the spatial and temporal modelling the best I could. In the end, simply including a bunch of categorical variables (for dates) and cluster information + lat/lon resulted in a very strong model (even without DEM or satellite, these models would score in top 10).

The rest of the features I just did experimentation and some hyper-parameter tuning over the course of the competition to get the best results.

10. *How does your model's performance vary?*

So a big factor here is that the regression modelling approach, it shrinks predictions toward the mean, so it is not necessarily very discriminatory to get many 5's.

11. *How did you evaluate performance of the model*

Since the model test set is held out in spatial blocks, I fit a selection model, and used this to hypertune the model results (limiting the results to not overfit models). So the test set is randomly sampled, but giving higher selection probabilities to areas with a higher probability in the train set of “being similar to” the held out set.

This resulted in much closer train/test splits compared to the real leaderboard data. You can see my discussion/example of that in the model_strategy notebook.

12. *Model simplifications?*

The model is overall quite fast. I believe it would be easier to use the NOAA lakes data to actually filter the sentinel imagery to the waterbody location (instead of segmenting the images). Also just caching the imagery data locally would save time over grabbing it over the internet.

13. What else did you try?

One thing I tried was using space-time lags of prior severity. But this tended to overfit in the model results.

14. New methods/techniques?

I suspect there is better solutions taking the satellite imagery data and identifying blooms. The solution here mostly leverages space/time autocorrelation to get decent results, so in practice will predict large areas just based on nearby samples.

This would be made much easier though if you had as inputs the locations of outlined water-bodies, as opposed to needing to try to pull out the water bodies from the images themselves.