Estimation of Daily Water Temperature using Random Forest

B Steele

September 29, 2023

- how do I establish a baseline for this? better than yesterday-is-today? (when I don't have a daily timeseries, can I do this?) – maybe try a global daily average of the last 10y or something? Hopefully this will create a near-daily TS and can go from there.

GH Repo

Scientific motivation and problem statement:

Water temperature is often an indicator of water quality (cite). Active monitoring of lakes, especially those that are difficult to access by monitoring personnel, is difficult. Additionally, manual monitoring of waterbodies (by physically visiting a site) and sensor networks to monitor water temperature, are costly endeavors (cite).

In this example, I will use Random Forest to estimate water surface temperature for reservoirs with long manual monitoring data from Northern Water. The features that I will be using to estimate surface temperature include summary NLDAS meteorological data (air temperature, precipitation, solar radiation, and wind) as well as static values for each of the reservoirs (elevation, surface area, maximum depth, volume, and shoreline distance).

The comparitive baseline

In addition to the manual sampling record that is maintained by Northern Water (n = 1125), I will be leveraging surface temperature estimates from the Landsat constellation, Landsat 4-9 (n = 5039. These thermal estimates are well-aligned with the manual monitoring data for the 7 reservoirs and have been bias-corrected for over estimates in the warmest months. 'Surface temperature' in the manual sampling record for this example is any measured temperature at >= 1m depth. I retain only the top-most value for temperature. Static variables are only available for 6 of 7 reservoirs, so Windy Gap reservoir has been dropped from this analysis.

[[add units to table]]

Table 1: Static variables used in the Random Forest algorithm. Windy Gap Reservoir has incomplete data and has been dropped from this analysis.

feature	elevation	area	shoreline_length	max_depth	volume
Carter Lake	5760	1100	12.0	180	112230
Grand Lake	8370	507	4.5	389	68621
Granby Reservoir	8280	7256	40.0	221	530000
Horsetooth Reservoir	5430	1850	25.0	200	150000
Shadow Mountain Reservoir	8367	1346	8.0	24	16800
Willow Creek Reservoir	5400	303	7.0	124	10500
Windy Gap Reservoir	NA	NA	NA	NA	445

Ideally, implementation of this algorithm will include application to lakes that have only Landsat-derived temperature estimates and that are outside of this dataset. Because I want this algorithm to perform well on new lakes, I want to take steps to make sure that it is not overfit to these specific lakes.

No pre-processing was completed for these data for regularization, as decision trees make purely empirical decisions, and that type of pre-processing is not usually necessary. I have pre-processed the NLDAS data to provide summaries of the previous day weather, 3 days prior, and 5 days prior - meaning, the model does not use *today's* weather for prediction.

Training/Validation/Testing

- leave one out by lake – look at Willard et al for their cv method – if the cv's are all performing reasonably, combine all data, model and use to forecast – if the cv's perform differently, consider an ensemble method
- for 'test' use the hindcasting!

– we're going to do leave-one-out train-val and then the test will be on the hindcasted data

It's clear that there are site-level differences in temperature range and general seasonal response (fig?). These differences are likely due to static variables that differentiate these lakes. That said, if I add in site-level information, the algorithm will quickly learn those key attributes and likely overfit to the data, not allowing for generalization beyond these lakes.

For testing, I'll be using timeseries-aware k-fold training and validation sets, splitting the data in 5-year chunks since 2008, resulting in 4 train-test iterations.

```
from graphviz import Source
```

```
tree = rf.estimators_[29]
filename = 'RF_temp_tree'
local_path = 'RandomForest/figures'
# Export the image to a dot file
export_graphviz(tree, out_file = '/Users/steeleb/Documents/GitHub/ATS-ML-Fall2023/{}/{}.do
Source.from_file('/Users/steeleb/Documents/GitHub/ATS-ML-Fall2023/{}/{}.dot'.format(local_
```

```
<graphviz.sources.Source object at 0x2e43dd240>
```

Results

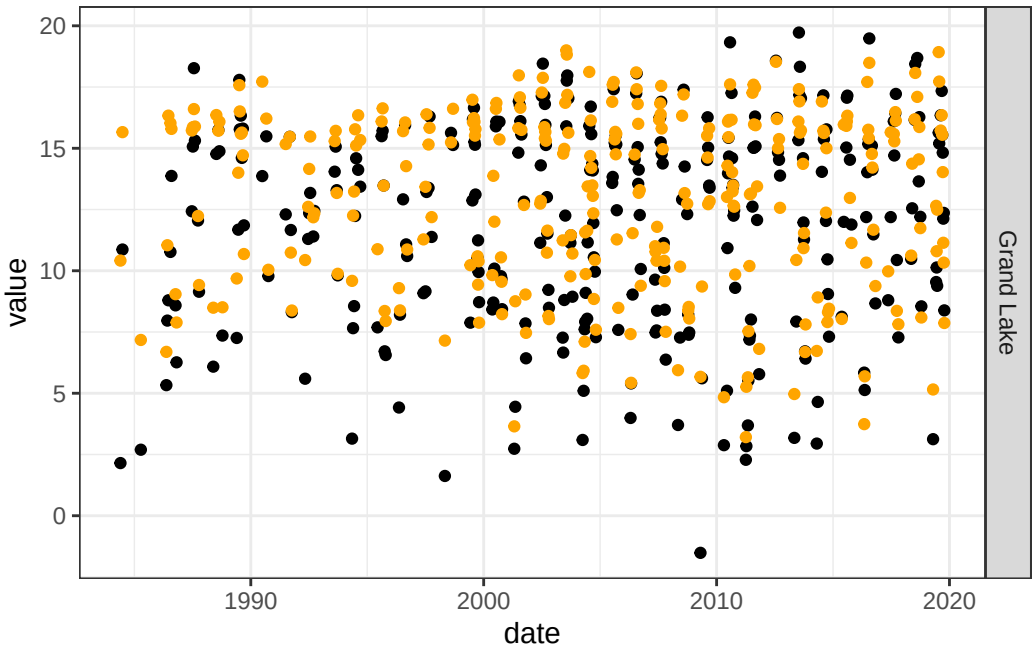

Figure 1: Datetime graph with actual values (black) and predicted values (orange), which shows the model is capturing the dirunal cycle.

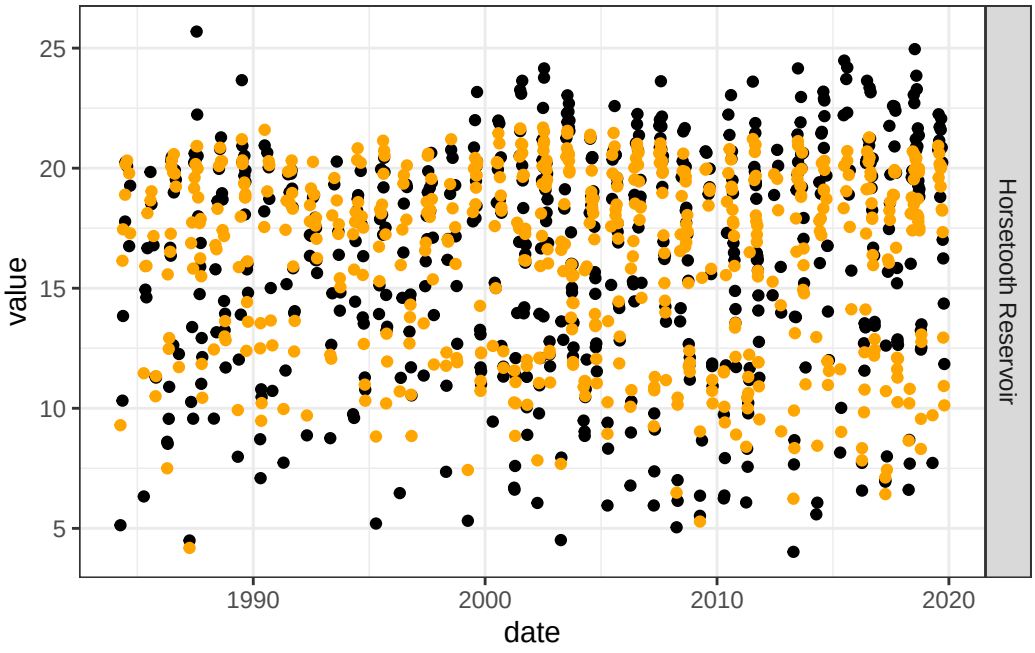

Figure 2: Datetime graph with actual values (black) and predicted values (orange), which shows the model is capturing the dirunal cycle.

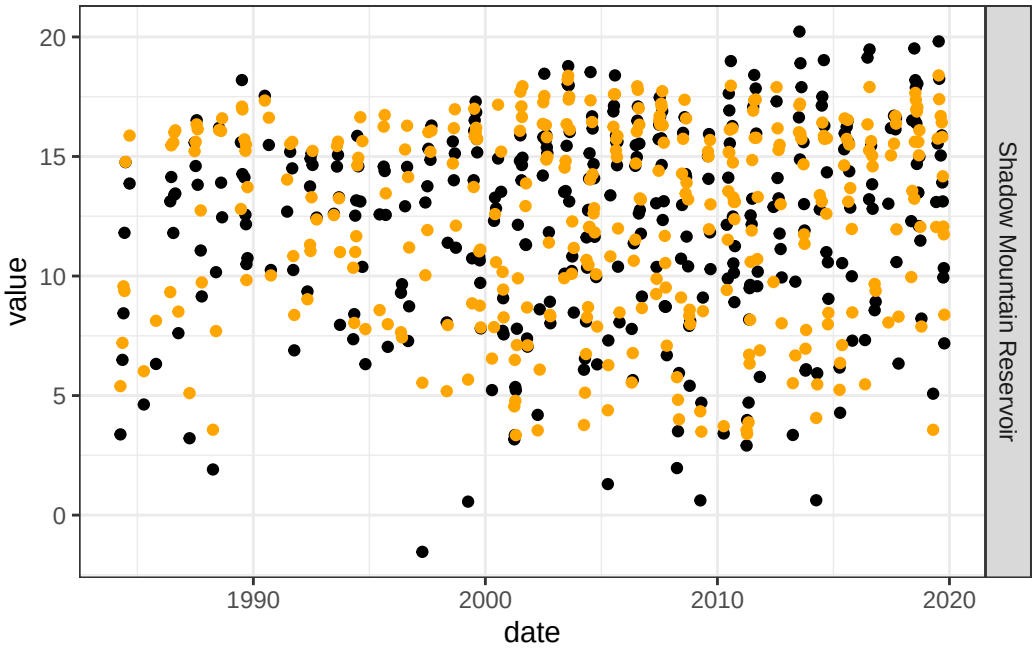

Figure 3: Datetime graph with actual values (black) and predicted values (orange), which shows the model is capturing the dirunal cycle.

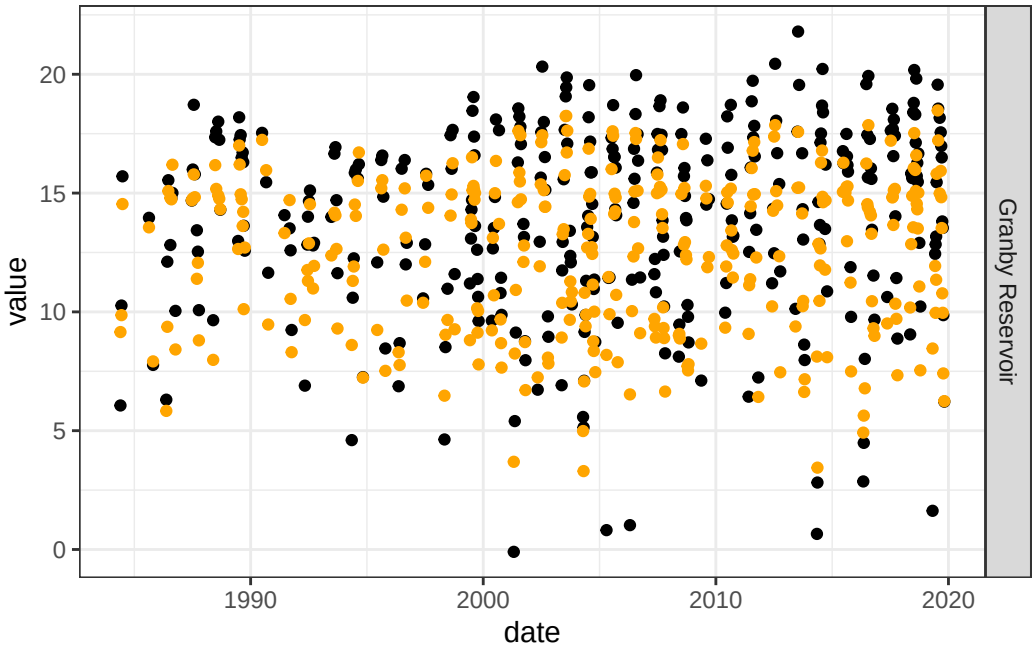

Figure 4: Datetime graph with actual values (black) and predicted values (orange), which shows the model is capturing the dirunal cycle.

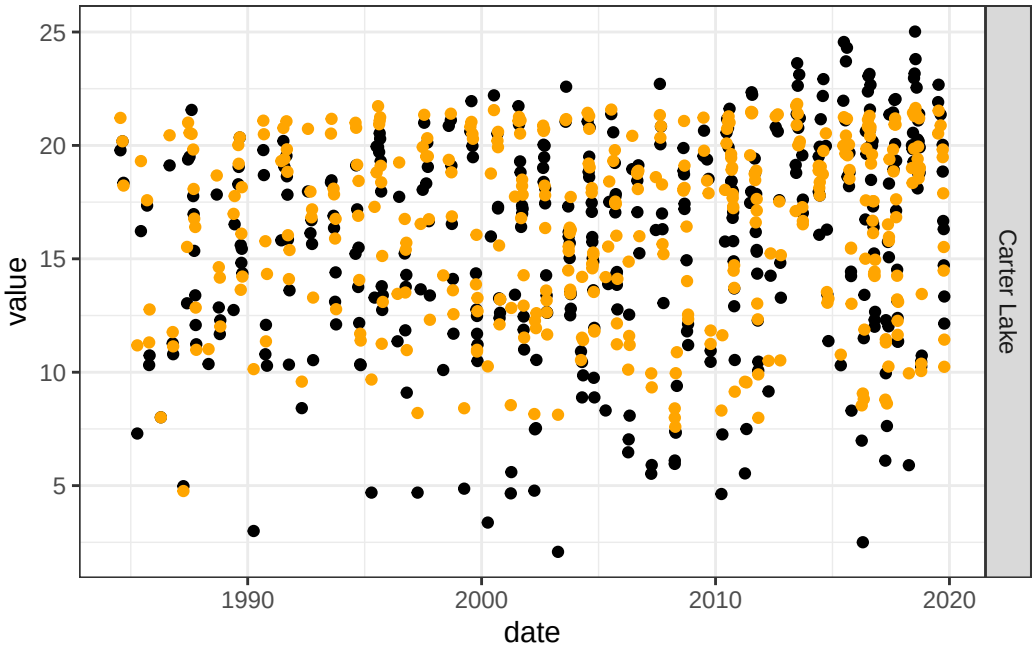

Figure 5: Datetime graph with actual values (black) and predicted values (orange), which shows the model is capturing the dirunal cycle.

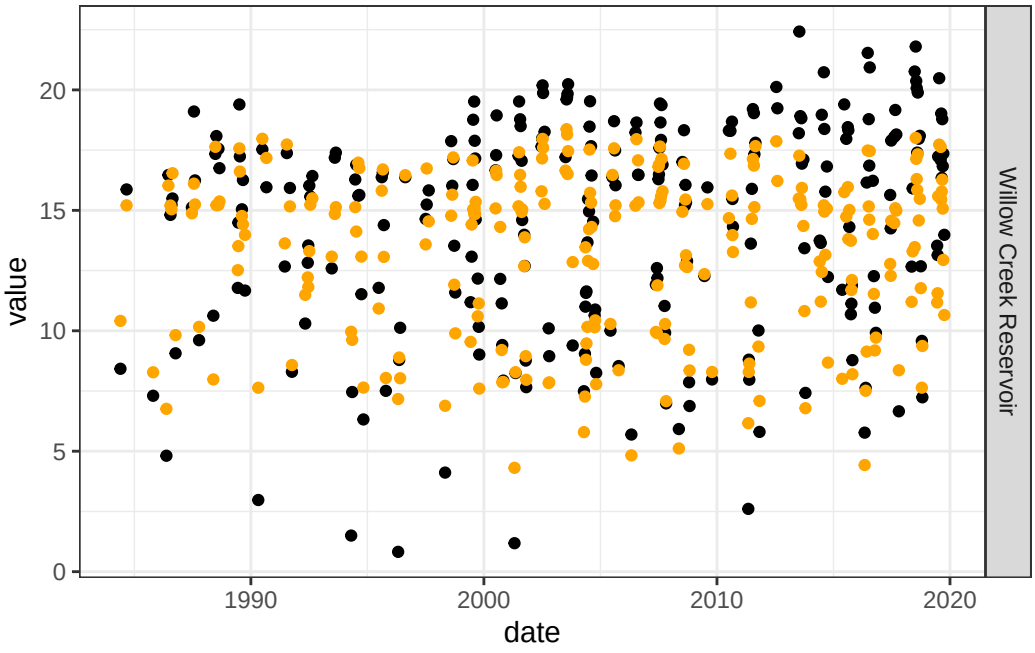

Figure 6: Datetime graph with actual values (black) and predicted values (orange), which shows the model is capturing the dirunal cycle.

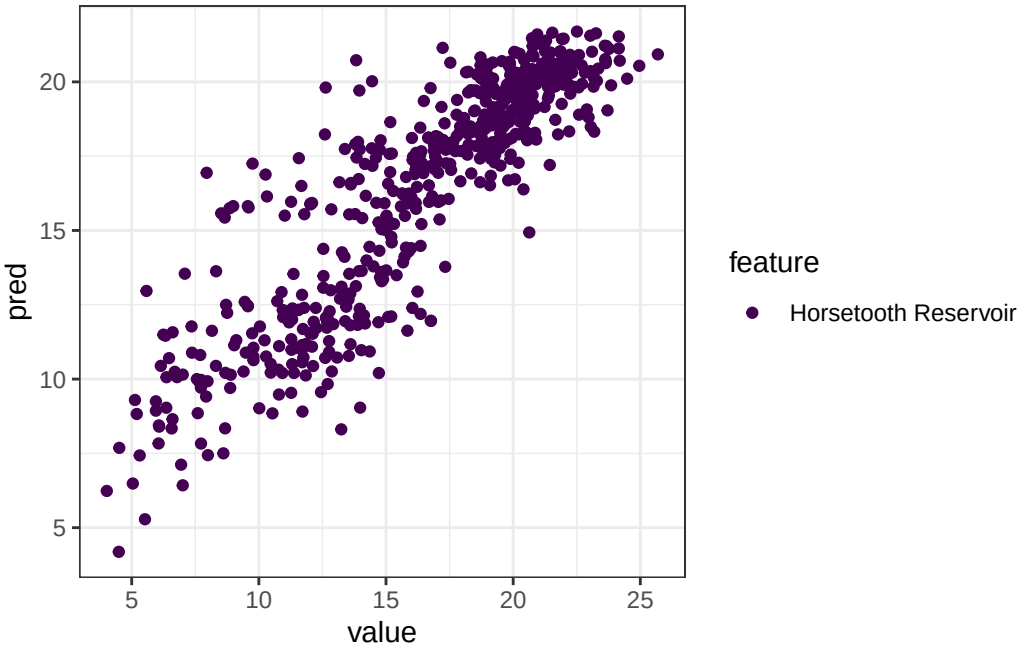

Figure 7: Scatter plot of predicted temperature and observed temperature at each of the 6 lakes in the dataset.