Estimation of Daily Water Temperature using Neural Networks

B Steele

October 15, 2023

GH Repo

Questions:

- I'd like the train/val figures to be plotted as MSE in degrees, not transformed values. If I use a Normalization layer within the DNN instead of the sklearn pipeline, will these show up as 'real values'? Did I just end up at a dead end for inverse processing of those values?
- I don't think that the features I am using are sufficient data for this method. I think this is likely an 'I need better features' (and likely I need more data) problem
- Can I use SHAP on continuous data? If so, would I apply it to the whole model (if it was well-performing?) instead of by 'correct' assignments of category like in the example code here?

Scientific motivation and problem statement:

Water temperature is often an indicator of water quality, as it governs much of the biological activity in freshwater. While temperature is an important parameter to monitor in freshwater lakes, manual monitoring of waterbodies (by physically visiting a site) and sensor networks to monitor water temperature, are costly endeavors.

In this example, I will use a simple fully-connected neural network to estimate water surface temperature for reservoirs with long manual monitoring data from Northern Water, the municipal subdistrict that delivers drinking water to approximately 1 million people in northern Colorado and irrigation water for ~600,000 acres of land. 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). To capture seasonal

dynamics, I've also included the numerical day of year to the feature set. The NLDAS data have been summarized for the previous day's weather, 3 days prior, and 5 days prior - meaning, the model does not use *today's* weather for prediction.

The comparative baseline for this analysis will be the day-of-year average water temperature across all lakes and years, where there are at least 3 values contributing to the mean. The baseline estimates result in a MAE of 2.25 deg C and MSE of 2.75 deg C.

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.

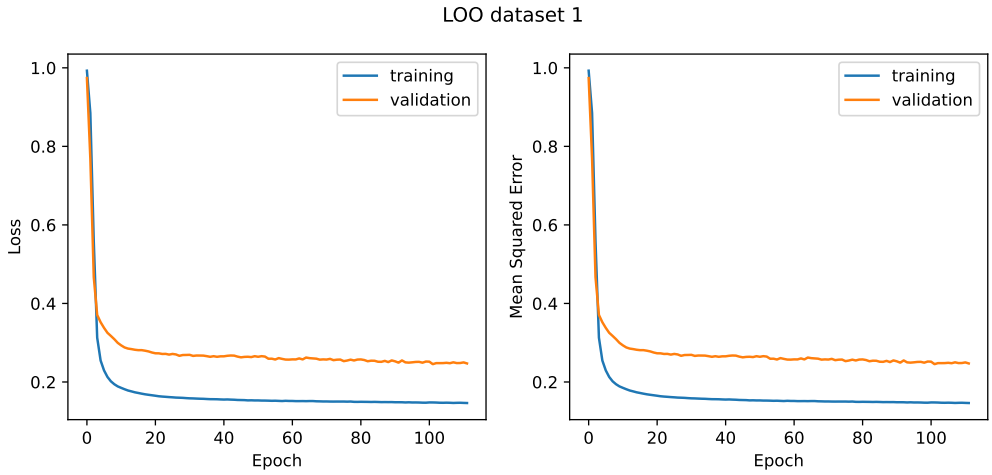
All precipitation data are right skewed heavily biased to low precip values including zero, to account for this and make the distribution more normal, I added 0.001 to each value and applied a square root transformation to this subset. The wind data were left skewed and to transform the distribution, I used a log function. All features and inputs were then scaled using the `StandardScaler()` function to get the values closer to zero, which are preferable for neural networks. These transformations were completed using the `make_pipeline()` function from the sklearn package.

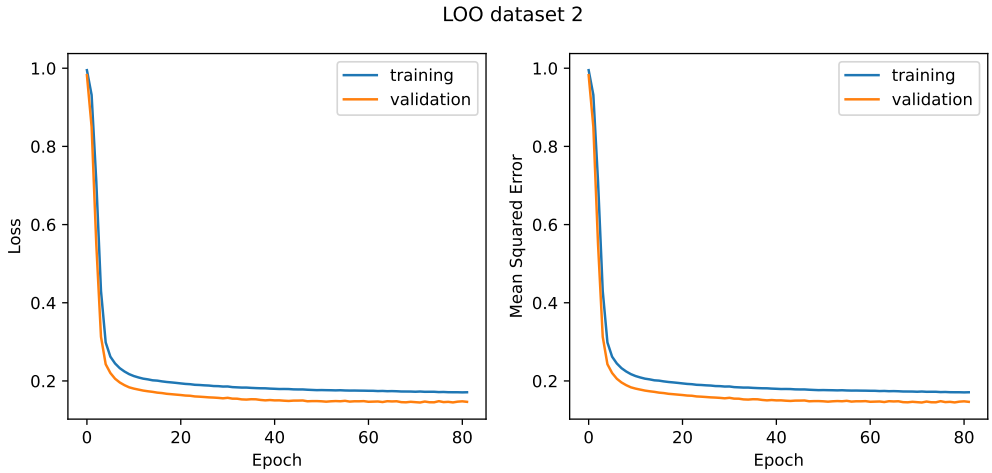
Training/Validation/Testing

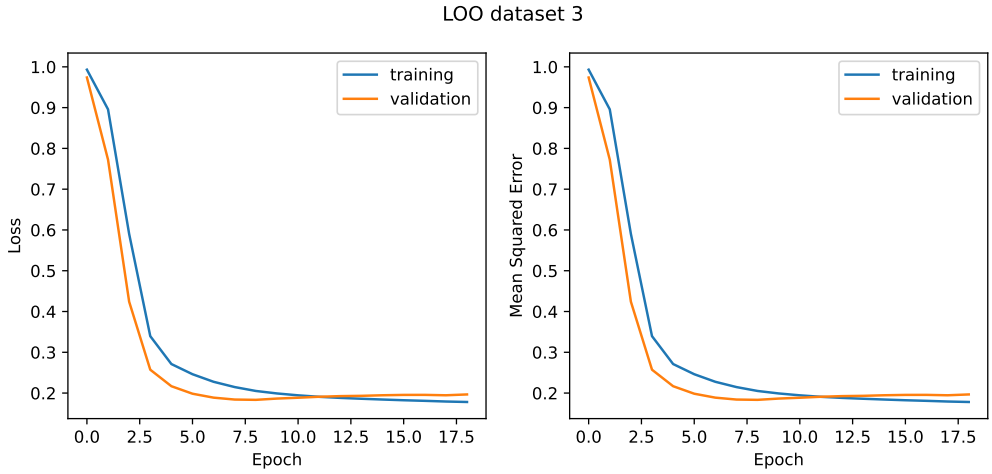
Eventual implementation of this algorithm will include forecasting of temperature for these lakes as well as lakes that have only Landsat-derived temperature estimates and that are not included in this dataset. Because I want this algorithm to perform well on new lakes, I want to take steps to make sure that the algorithm is not overfit to these specific lakes static characteristics. While this information may be important for alogorithm development, the model may have a propensity to "learn" those key attributes and overfit to the data, not allowing for generalization beyond these lakes.

For training and validation I will use two techniques. First, a leave-one-out method that will result in six NN models where each iteration will use data from a single lake for validation and the other five for training. Because the random forest models did not appear to overfit to the static variables, I'm also trying a timeseries method that will subset the data into ~10 year increments and leave one increment out per training and use it for validation per iteration. Since the intended implementation will be daily forecasts, testing performance will be assessed through hindcasting. The hindcast dataset is a holdout dataset beginning in 2021 across all lakes.

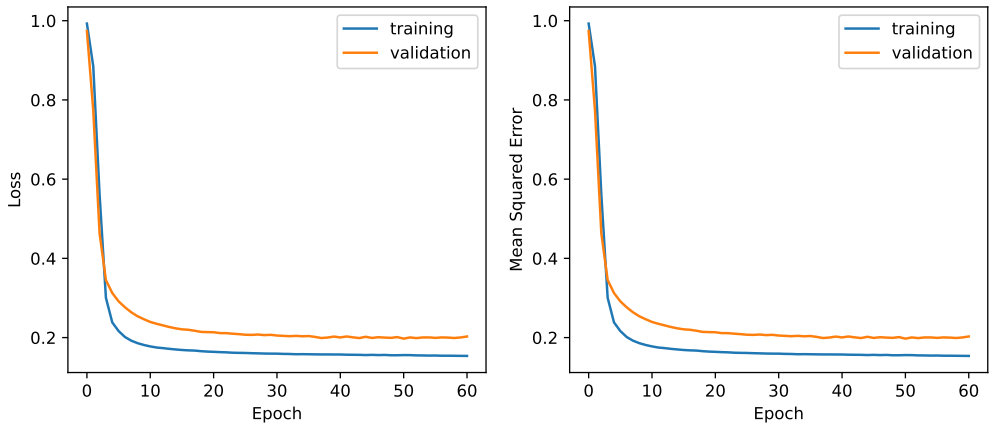
Results


LOO dataset 1
training
validation
Loss
Mean Squared Error
Epoch

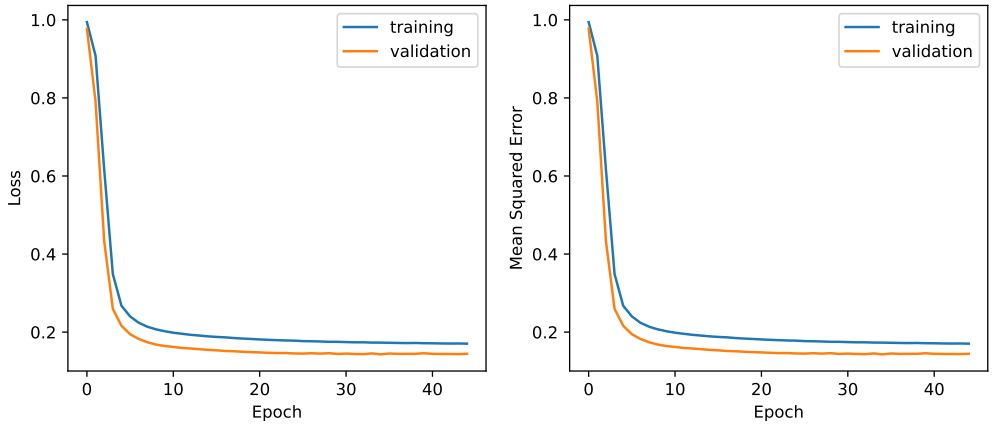

LOO dataset 2
training
validation
Loss
Mean Squared Error
Epoch


LOO dataset 3
training
validation
Loss
Mean Squared Error
Epoch

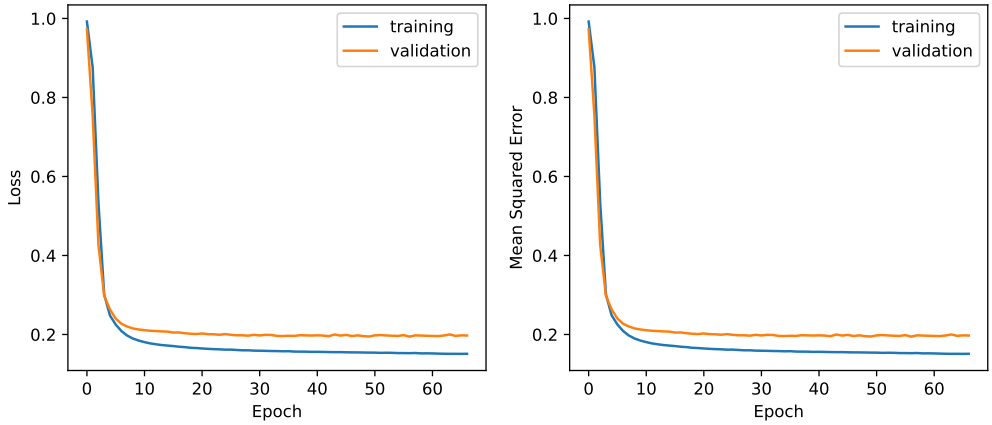
LOO dataset 4



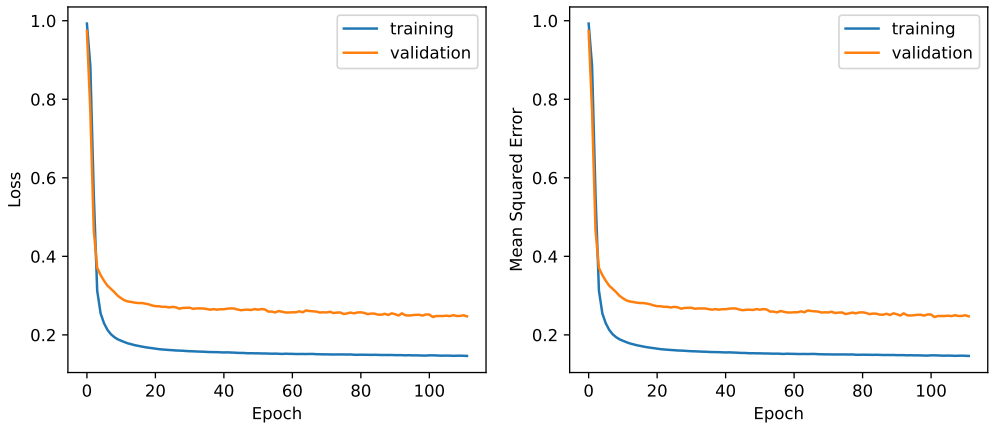
LOO dataset 5



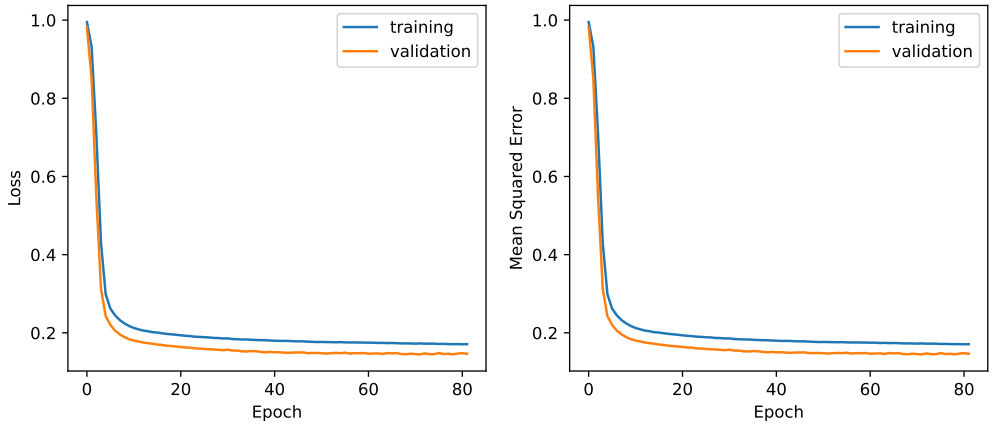
LOO dataset 6



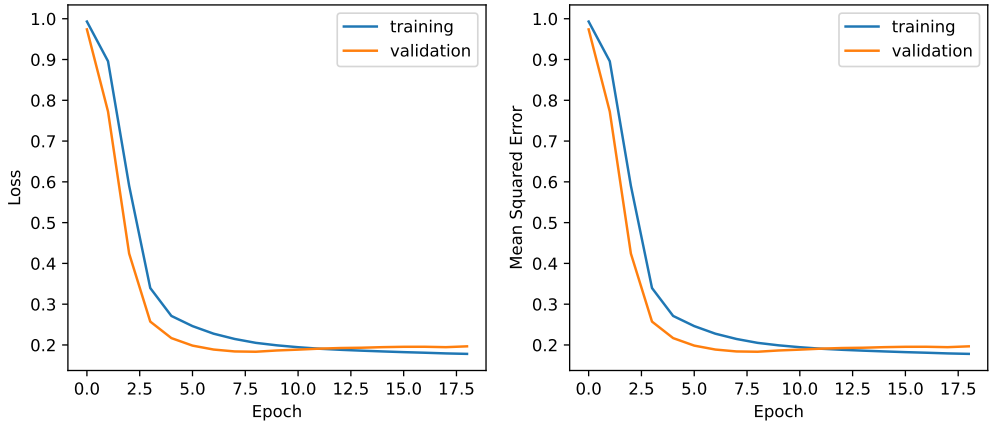
TS dataset 1

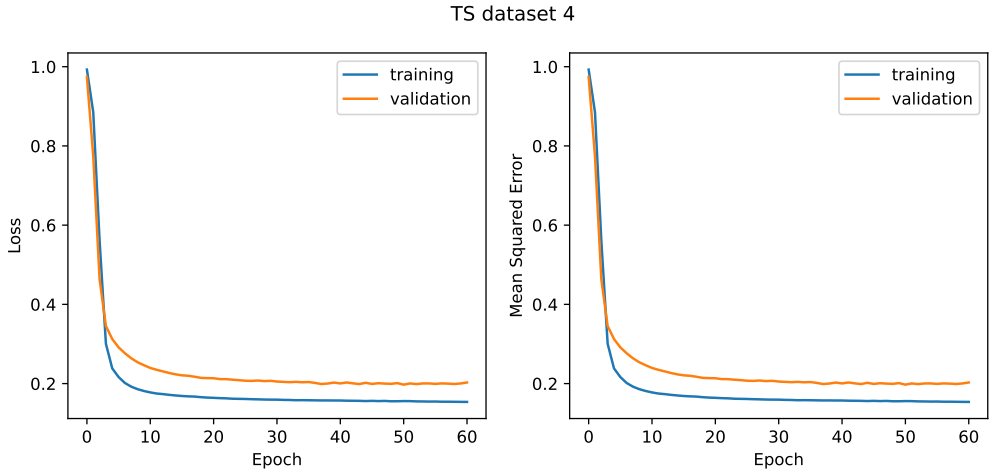


TS dataset 2



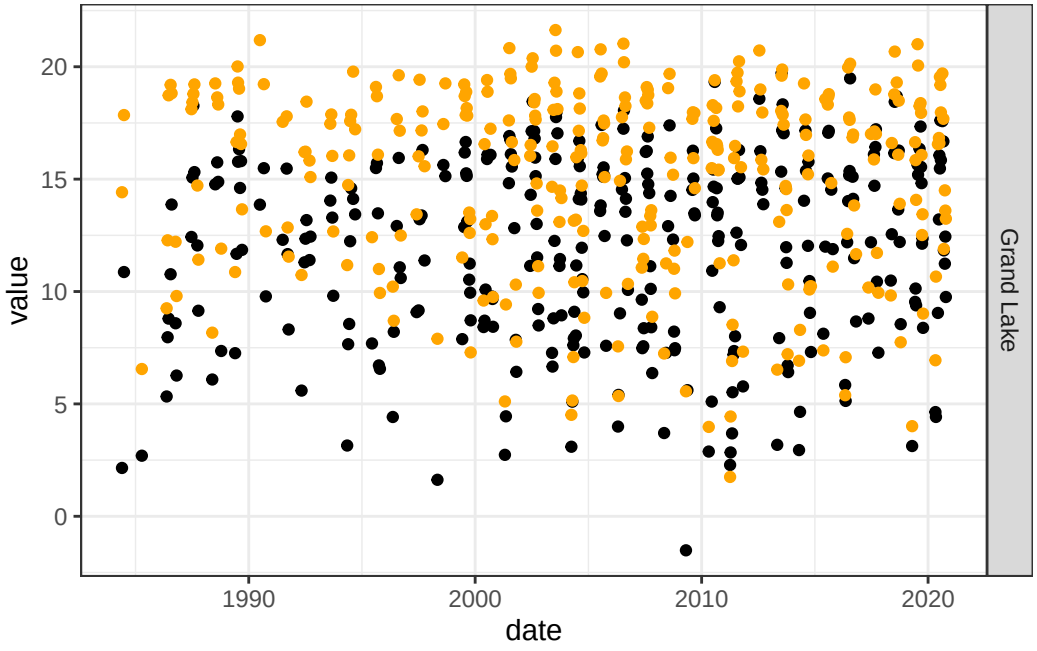
TS dataset 3

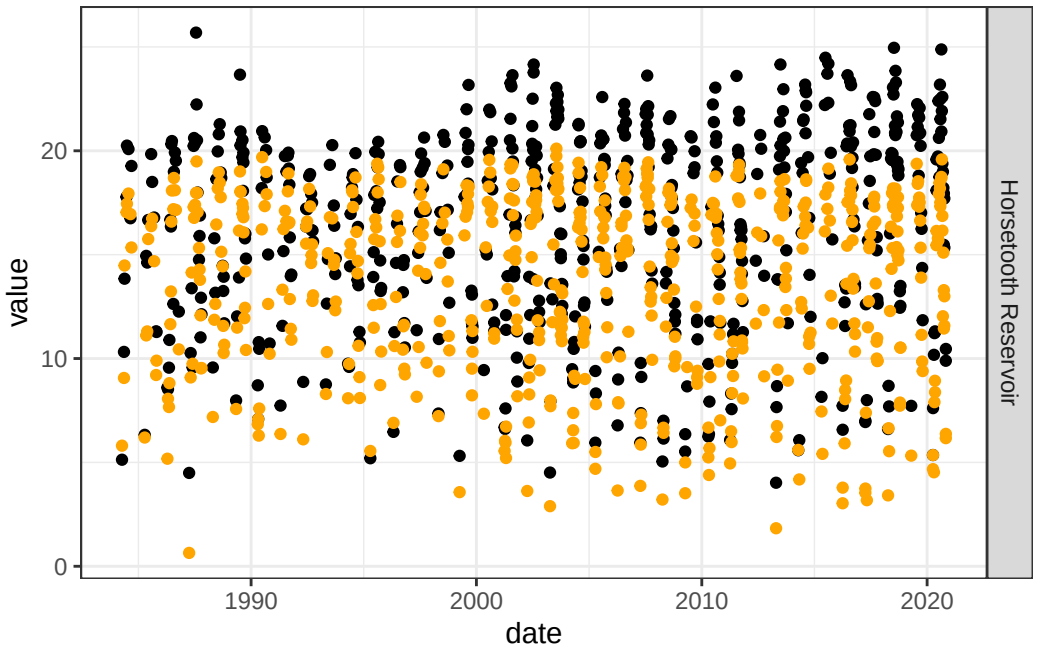


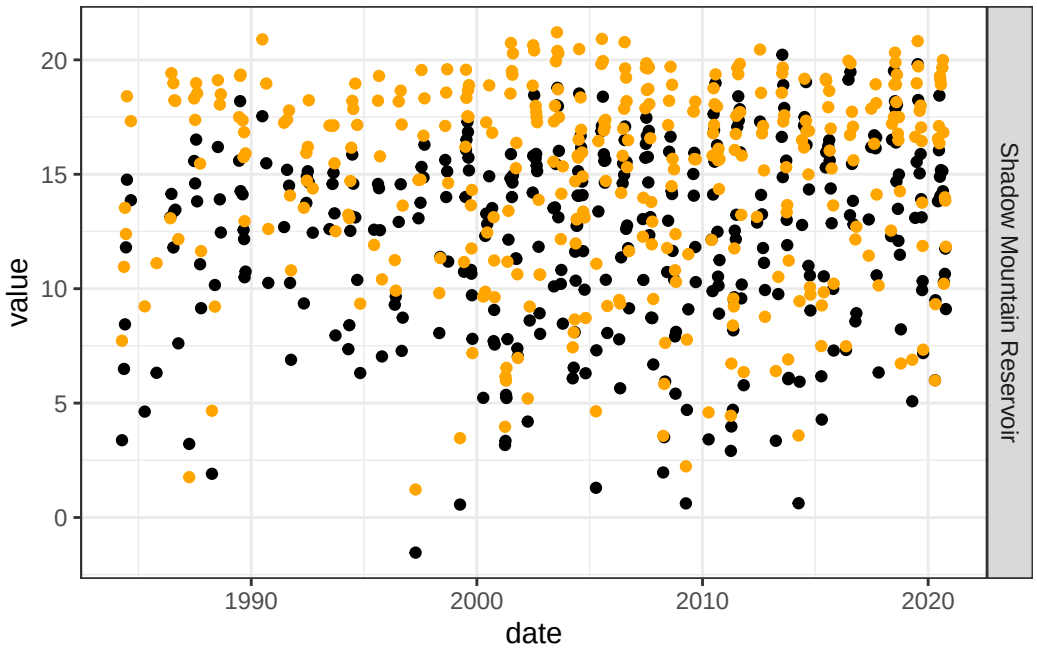


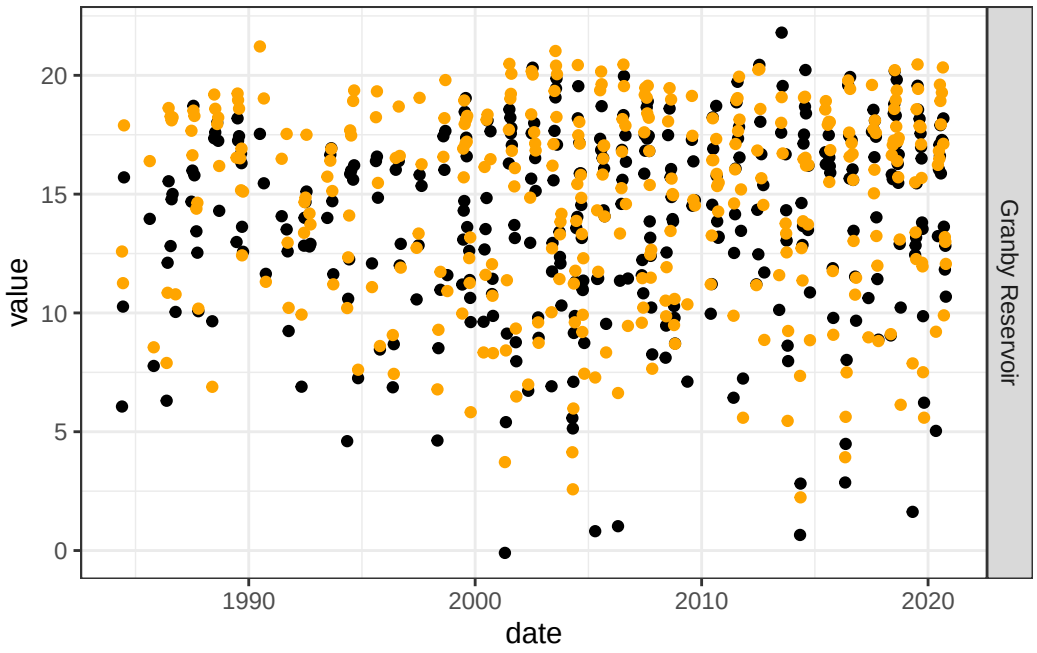
Leave one out validations

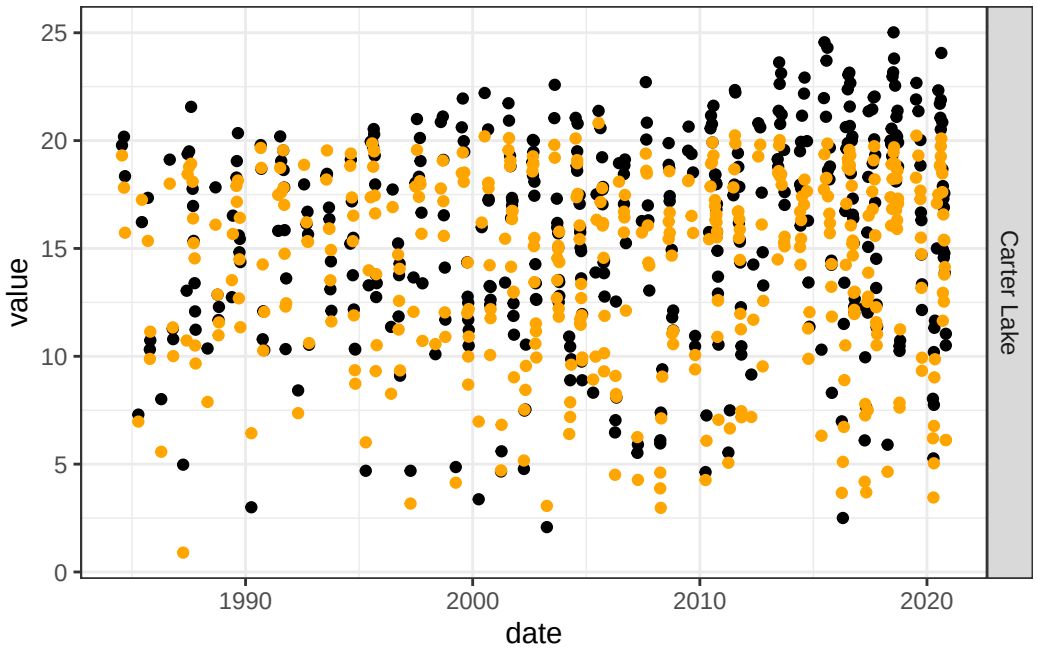
These are super hit-or-miss. MSE ranges from 3-10. Pretty terrible.

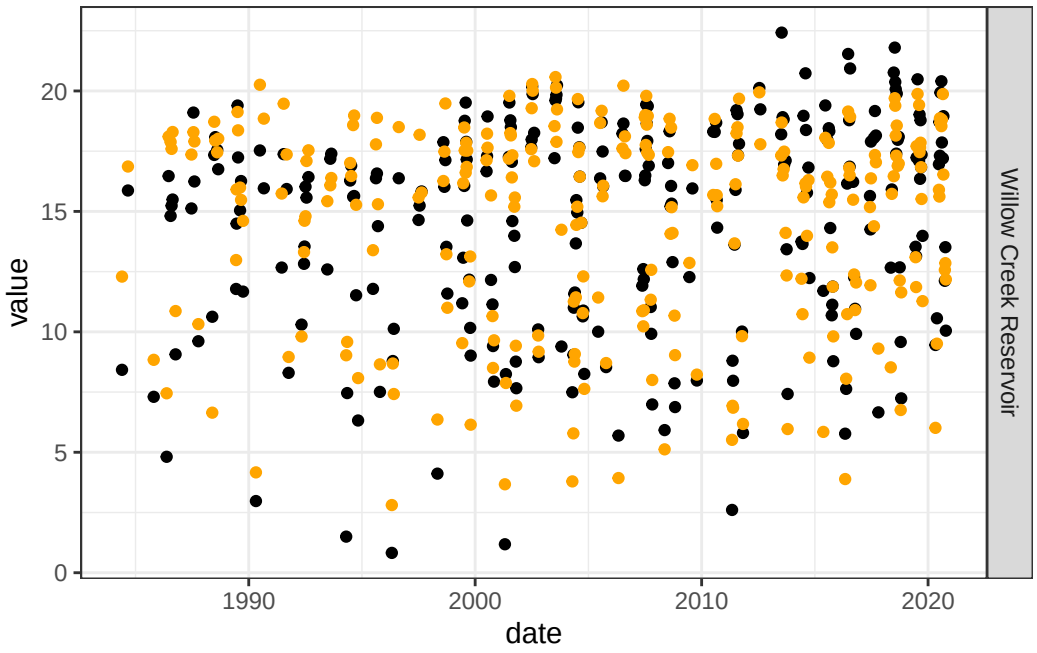



value
20
10
0
1990
2000
2010
2020
date
Horsetooth Reservoir


value
20
15
10
5
0
1990
2000
2010
2020
date
Shadow Mountain Reservoir

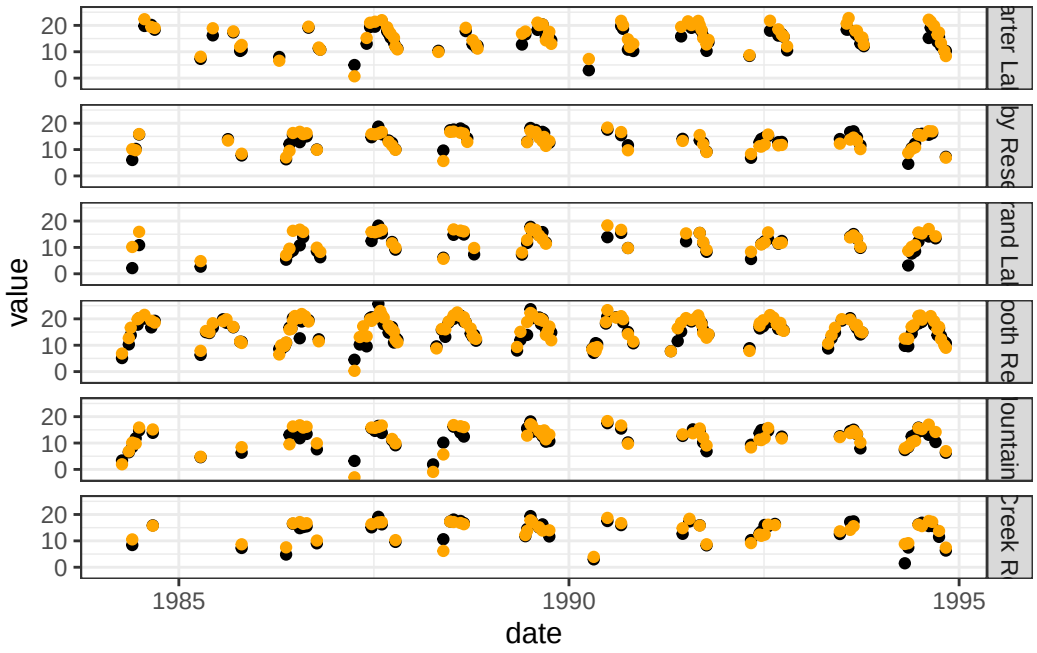

value
20
15
10
5
0
1990
2000
2010
2020
date
Granby Reservoir

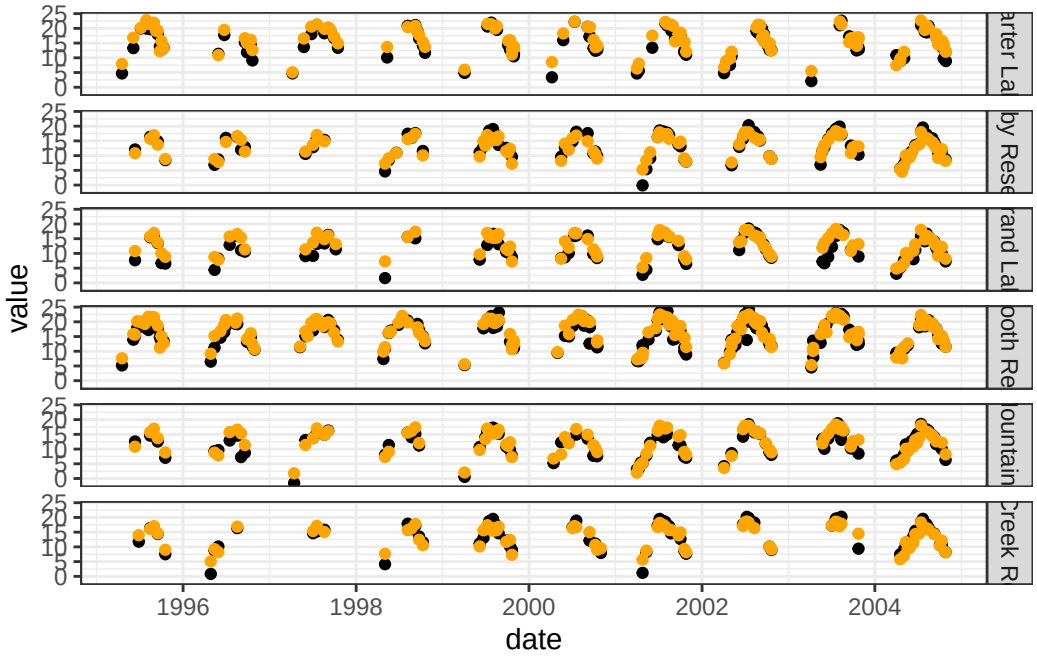

value
25
20
15
10
5
0
1990
2000
2010
2020
date
Carter Lake

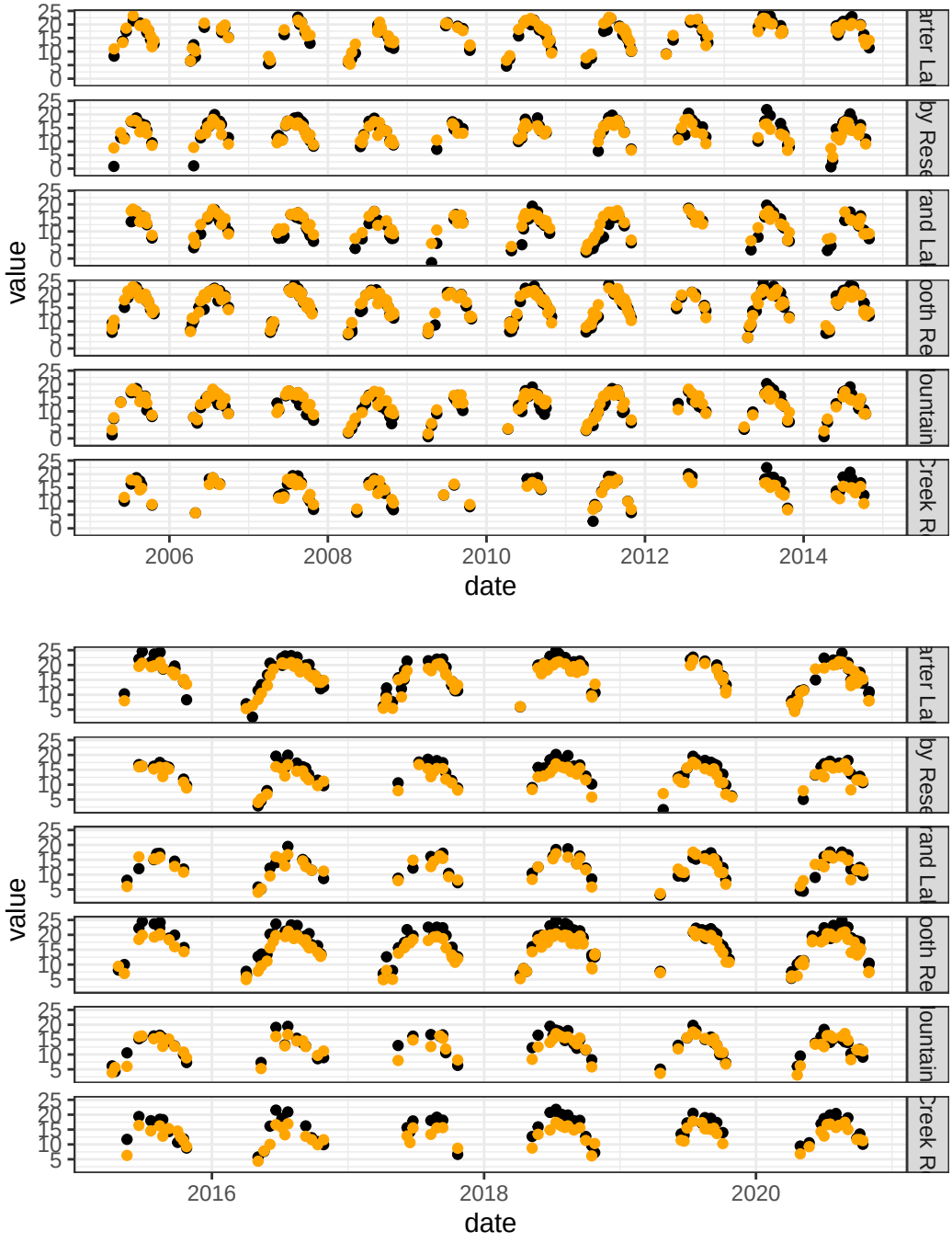


Timeseries split validations

These are good until the most recent data, which look horrid. MSE is smaller, but still higher than baseline (3-5).


value
20
10
0
1985
1990
1995
date


value
25
20
15
10
5
0
1996
1998
2000
2002
2004
date



Hyper-parameter tuning

Current settings:

```
settings = {
    "hiddens": [3, 3],
    "activations": ["relu", "relu"],
    "learning_rate": 0.001,
    "random_seed": 57,
    "max_epochs": 1000,
    "batch_size": 32,
    "patience": 10,
    "dropout_rate": 0,
}
```