Water Temperature Prediction

Day of Year and Depth XGBoost Model

Gregory Harrison, Claire Oberg

Nov 28, 2022

Read in Current Data from Lake Sunapee

```
raw <- readr::read_csv("https://s3.flare-forecast.org/targets/sunp/sunp-targets-insitu.csv")
#raw <- readr::read_csv("sunp-targets-insitu.csv")
```

Convert Data to Dataframe

```
# Convert to Dataframe
df = data.frame(raw)

# Generate year and day of year from the date column
df[['year']] <- strptime(df[['date']], format = "%Y-%m-%d")$year + 1900
df[['dayofyear']] <- as.numeric( format(df[['date']], '%j'))

# Filter data according to our assumptions being:
# only calculating at noon
# years during and after 2011 and not including 2015 and 2017 for low amounts of data
# Day of year range only between 162 and 278
df <- df %>%
  filter(hour == 12) %>%
  filter(year >= 2011) %>%
  filter(year != 2017) %>%
  filter(year != 2015) %>%
  filter(dayofyear >= 162) %>%
  filter(dayofyear <= 278)


#Remove the date, variable, and hour columns from our dataframe
df$date = NULL
df$variable = NULL
df$hour = NULL
```

Separate Training and Testing Sets

```
# Separate our data into training (all years but 2013 and 2022) and
# testings (years 2013 and 2022)
waterTrain = df[df$year != 2013 & df$year != 2022, ]
waterTest = df[df$year == 2013 | df$year == 2022, ]

# Drop Columns with NA's in them
waterTrain = na.omit(waterTrain)
waterTest = na.omit(waterTest)
tempWater = waterTrain
```

Convert Training and Testing set for XGBoost

```
# Remove the year columns now that we've used them to seperate the date
waterTrain$year = NULL
waterTest$year = NULL
```

```
# Generate our labels as the current water temperature
train.label = waterTrain$value
test.label = waterTest$value

# Remove the temperature values from the input data
waterTrain$value = NULL
waterTest$value = NULL

# Convert the input data to a matrix for xgboost
train.data = as.matrix(waterTrain)
test.data = as.matrix(waterTest)
```

Train Model

```
# Generate Training Input for XGBoost
dtrain<-xgb.DMatrix(data = train.data, label = train.label)
# Train our model
bst <- xgboost(data = dtrain, max.depth = 10, eta = 1, nthread = 2, nrounds = 20, verbose = 1)
```

```
[1] train-rmse:1.359545
[2] train-rmse:1.046646
[3] train-rmse:0.962883
[4] train-rmse:0.936954
[5] train-rmse:0.887578
[6] train-rmse:0.882127
[7] train-rmse:0.880201
[8] train-rmse:0.874991
[9] train-rmse:0.865292
[10]    train-rmse:0.860059
[11]    train-rmse:0.858022
[12]    train-rmse:0.854641
[13]    train-rmse:0.852518
[14]    train-rmse:0.849185
[15]    train-rmse:0.848476
[16]    train-rmse:0.846940
[17]    train-rmse:0.846066
[18]    train-rmse:0.845080
[19]    train-rmse:0.844542
[20]    train-rmse:0.843993
```

```
# Product Predictions for our Testing Dataset
pred <- predict(bst, test.data)
# Calculate Mean Absolute Error
mean(abs(pred-test.label))
```

```
[1] 1.492315
```

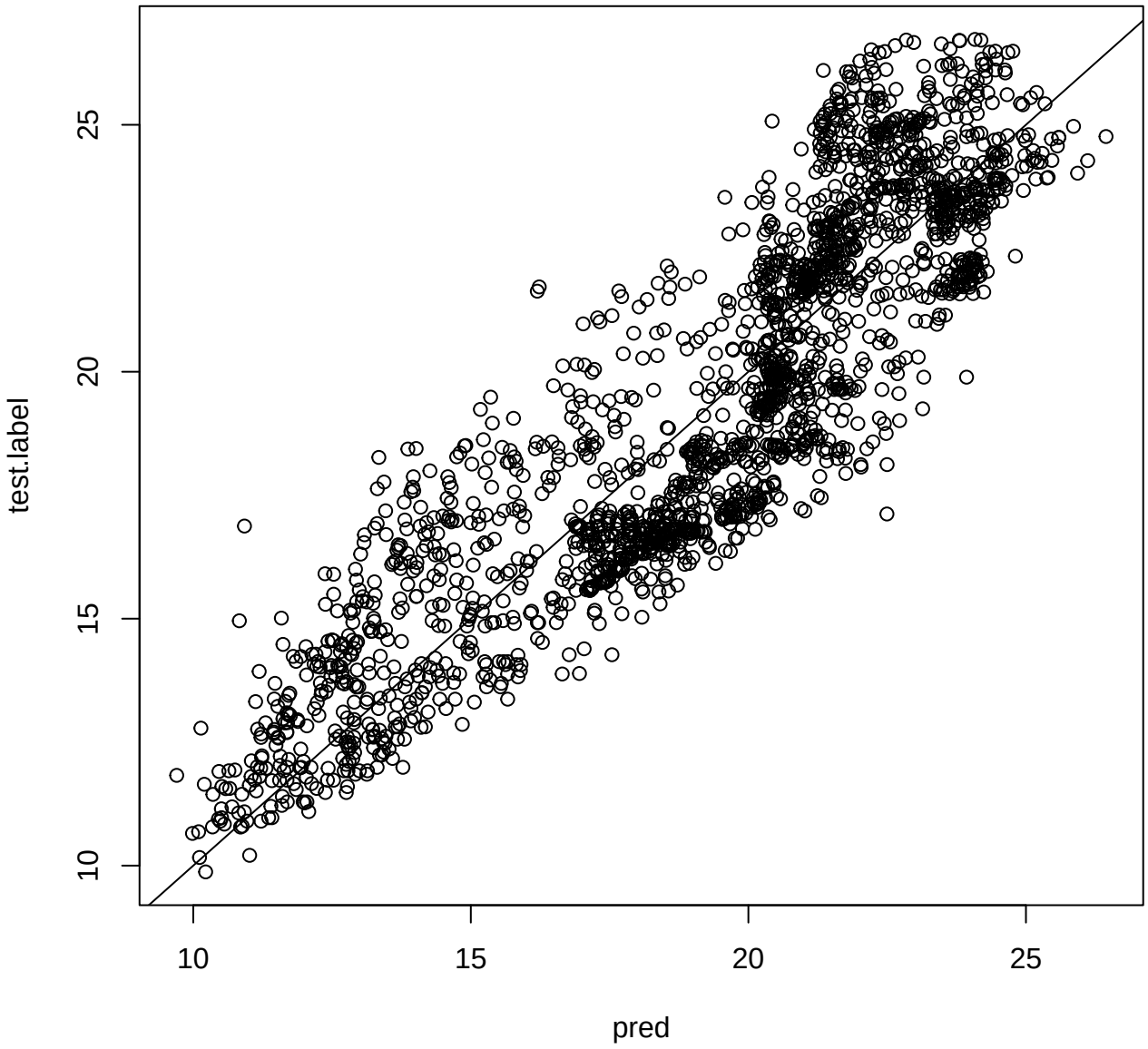
See how the model performs on the original training data.

```
predTrain <- predict(bst, train.data)
mean(abs(predTrain-train.label))
```

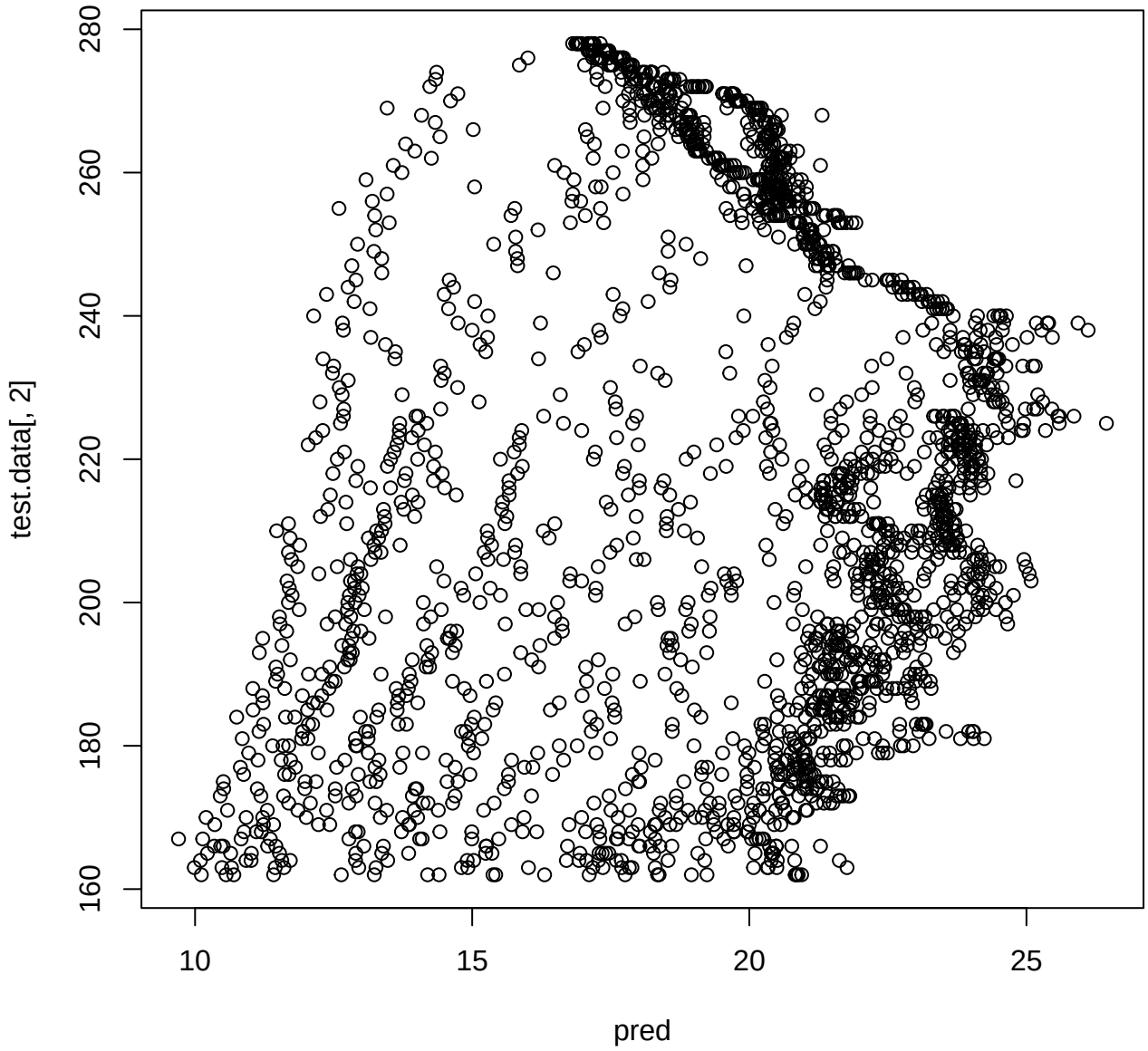
```
[1] 0.5711845
```

Generate Some Graph of our data Rename test.label observed

```
# Plot of Predicted vs. Actual Values
plot(pred, test.label)
abline(0,1)
```



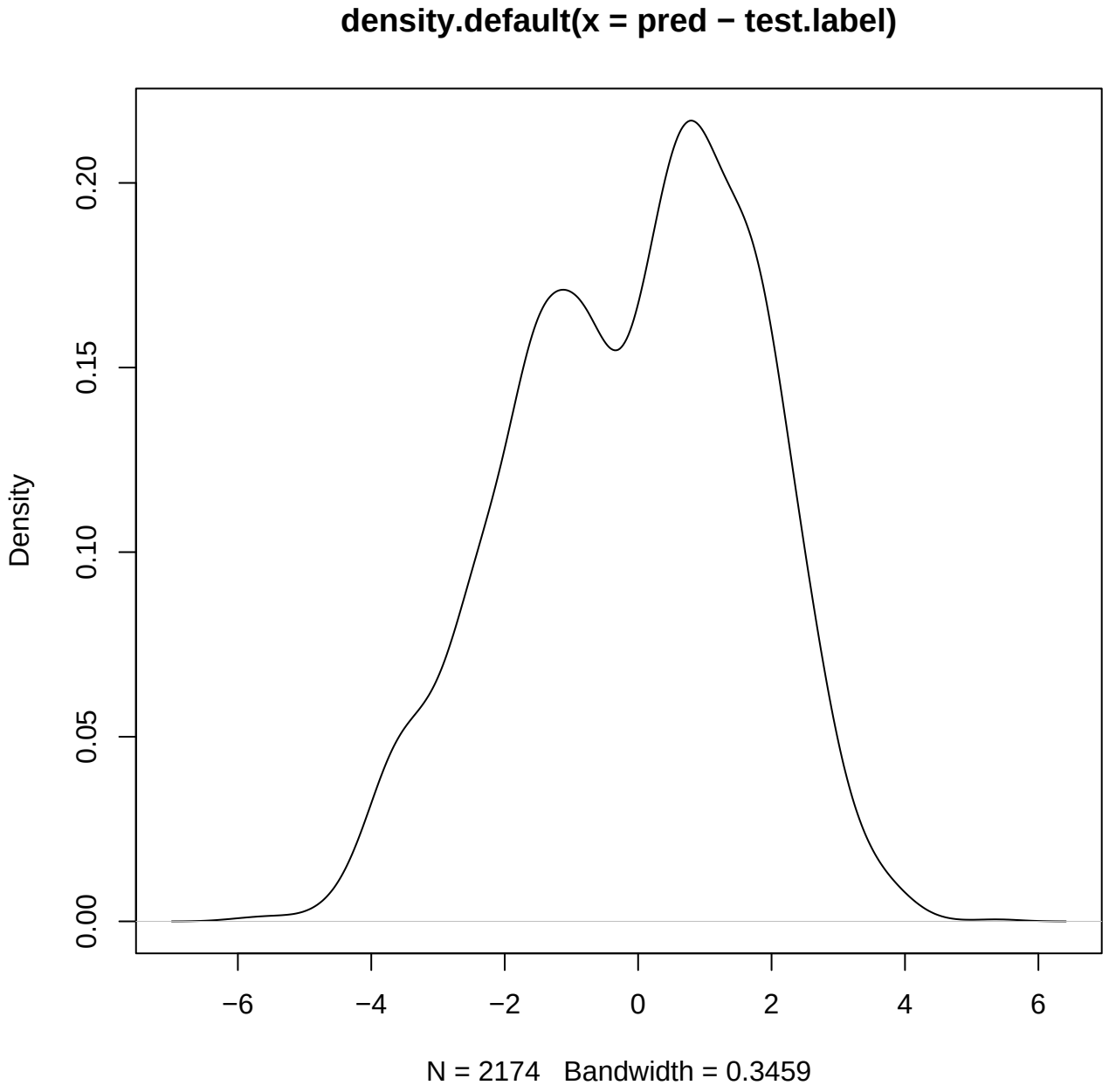

```
# Plot of Predicted vs. Day of Year
plot(pred, test.data[,2])
```



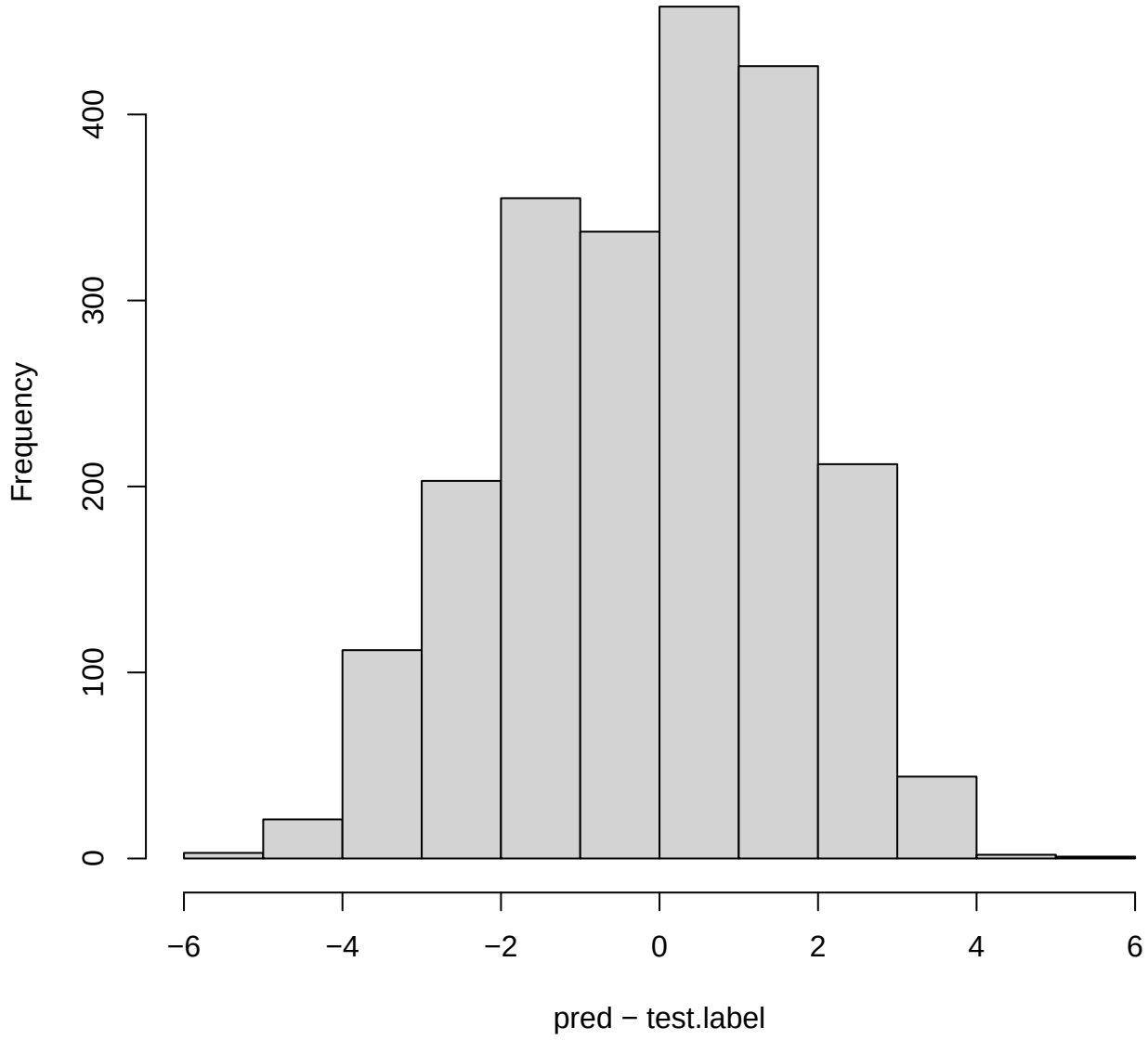

```
# Summary of Error
summary(pred-test.label)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-5.95282 -1.40996  0.14864 -0.06598  1.32030  5.37587
```

```
# Histogram of Error
den<-density(pred-test.label)
plot(den)
```




```
hist(pred-test.label)
```




```
sqrt(mean((pred-test.label)^2))
```

```
[1] 1.787911
```