Water Temperature Prediction

Day of Year, Depth, 7 day Previous Water Temperature, and Future Weather XGBoost Model

Gregory Harrison, Claire Oberg, Zihao Li

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

```
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'))


df <- df %>%
  filter(hour == 12) %>%
  filter(year >= 2011) %>%
  filter(year != 2017) %>%
  filter(year != 2015)

dfPrev = select(df, c("depth", "year", "dayofyear", "value"))
# 7 in line below denotes how many days previous to use in model
dfPrev["dayofyear"] = dfPrev["dayofyear"] + 7
dfPrev["previousvalue"] = dfPrev["value"]
dfPrev$value = NULL

dfNew = merge(df, dfPrev, by=c("year","dayofyear", "depth"))
```

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 <- dfNew %>%
  filter(dayofyear >= 162) %>%
  filter(dayofyear <= 278)
```

Weather:

```
setwd("/Users/eric/Desktop/Fall22/CMDA4864/Final_Project/")
weather <- read.csv("all_data.csv", stringsAsFactors = T)
head(weather)
```

```
             datetime location windDirectionInstantaneous_deg
1 2007-08-27 23:00:00     loon                             NA
2 2007-08-27 23:10:00     loon                             NA
3 2007-08-27 23:20:00     loon                             NA
4 2007-08-27 23:30:00     loon                             NA
5 2007-08-27 23:40:00     loon                             NA
6 2007-08-27 23:50:00     loon                             NA
  windSpeedInstantaneous_mps flag_winddir radiationIncomingPAR_umolm2s flag_par
1                         NA            e                           NA     <NA>
2                        1.6            e                            0     <NA>
```

```
3                        1.7            e                            0     <NA>
4                        1.9            e                            0     <NA>
5                        2.3            e                            0     <NA>
6                        1.7            e                            0     <NA>
  airTemperature_degC flag_airtemp windDirectionAverage_deg
1                  NA         <NA>                       NA
2               16.43         <NA>                       NA
3               16.37         <NA>                       NA
4               16.37         <NA>                       NA
5               16.09         <NA>                       NA
6               15.96         <NA>                       NA
  windSpeedAverage_mps flag_allwind relativeHumidity_perc flag_rh
1                   NA         <NA>                    NA    <NA>
2                   NA         <NA>                    NA    <NA>
3                   NA         <NA>                    NA    <NA>
4                   NA         <NA>                    NA    <NA>
5                   NA         <NA>                    NA    <NA>
6                   NA         <NA>                    NA    <NA>
  windGustSpeed_mps windGustDirection_deg
1                NA                    NA
2                NA                    NA
3                NA                    NA
4                NA                    NA
5                NA                    NA
6                NA                    NA
```

```r
# deal with datetime stuff
weather$date <- as.Date(weather$datetime)
weather[['dayofyear']] <- as.numeric( format(weather[['date']], '%j'))
weather$year <- as.numeric(format(weather$date,'%Y'))

# clean out the funky years etc, same way as cleaning water temps
clean_weather <- weather %>%
    filter(year >= 2011)

# cut out variables we won't use
small_weather <- clean_weather %>% select(17,18,19,8,11,13,6)

# calculate mins, maxes, and averages
calculated_weather <- aggregate(x = small_weather$airTemperature_degC,      # Specify data column
                         by = list(small_weather$date),                 # Specify group indicator
                         FUN = mean)
min_temp <- aggregate(x = small_weather$airTemperature_degC,
                         by = list(small_weather$date),
                         FUN = min)
max_temp <- aggregate(x = small_weather$airTemperature_degC,
                         by = list(small_weather$date),
                         FUN = max)
avg_rad <- aggregate(x = small_weather$radiationIncomingPAR_umolm2s,
                         by = list(small_weather$date),
                         FUN = mean)
avg_windspeed <- aggregate(x = small_weather$windSpeedAverage_mps,
                         by = list(small_weather$date),
                         FUN = mean)
avg_humidity <- aggregate(x = small_weather$relativeHumidity_perc,
                         by = list(small_weather$date),
                         FUN = mean)

# create and merge data frame
calculated_weather['min_airTemp'] <- min_temp$x
```

```
calculated_weather['max_airTemp'] <- max_temp$x
calculated_weather['avg_radian'] <- avg_rad$x
calculated_weather['avg_windspeed'] <- avg_windspeed$x
calculated_weather['avg_humidity'] <- avg_humidity$x
colnames(calculated_weather)[2] = "avg_airTemp"
colnames(calculated_weather)[1] = "date"

calculated_weather['year'] <- strptime(calculated_weather[['date']], format = "%Y-%m-%d")$year + 1900
calculated_weather['dayofyear'] <- as.numeric( format(calculated_weather[['date']], '%j'))

avg_airTemp_35 = rollapply(calculated_weather$avg_airTemp, width = 35, by=1, FUN = mean, na.rm=TRUE, align="le
calculated_weather['avg_airTemp_35'] = append(rep(NA,34), avg_airTemp_35)

min_airTemp_35 = rollapply(calculated_weather$min_airTemp, width = 35, by=1, FUN = mean, na.rm=TRUE, align="le
calculated_weather['min_airTemp_35'] = append(rep(NA,34), min_airTemp_35)

max_airTemp_35 = rollapply(calculated_weather$max_airTemp, width = 35, by=1, FUN = mean, na.rm=TRUE, align="le
calculated_weather['max_airTemp_35'] = append(rep(NA,34), max_airTemp_35)

avg_radian_35 = rollapply(calculated_weather$avg_radian, width = 35, by=1, FUN = mean, na.rm=TRUE, align="left
calculated_weather['avg_radian_35'] = append(rep(NA,34), avg_radian_35)

calculated_weather <- calculated_weather %>%
    filter(year >= 2011) %>%
    filter(year != 2017) %>%
    filter(year != 2015) %>%
    filter(dayofyear >= 162) %>%
    filter(dayofyear <= 278)
```

Merge:

```
merged <- merge(df, calculated_weather, by = 'date', all=TRUE)
df = merged

#Remove the date, variable, and hour columns from our dataframe
df$year = df$year.x
df$dayofyear = df$dayofyear.x
dfReduced = select(df, c("depth", "year", "dayofyear", "value", "avg_airTemp_35", 'min_airTemp_35', 'max_airTe

df = dfReduced

df = df[!is.na(df$avg_airTemp_35)&!is.na(df$min_airTemp_35)&!is.na(df$max_airTemp_35)&!is.na(df$avg_radian_35)
```

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 = 0.3, nthread = 2, nrounds = 20, verbose = 1)
```

```
[1] train-rmse:13.796620
[2] train-rmse:9.691563
[3] train-rmse:6.820812
[4] train-rmse:4.817314
[5] train-rmse:3.421509
[6] train-rmse:2.452971
[7] train-rmse:1.778972
[8] train-rmse:1.309549
[9] train-rmse:0.982108
[10]    train-rmse:0.754031
[11]    train-rmse:0.601456
[12]    train-rmse:0.495170
[13]    train-rmse:0.425865
[14]    train-rmse:0.382870
[15]    train-rmse:0.345977
[16]    train-rmse:0.324872
[17]    train-rmse:0.306893
[18]    train-rmse:0.292003
[19]    train-rmse:0.278809
[20]    train-rmse:0.266415
```

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

```
[1] 0.8349715
```

```
# Calculate Root Mean Squared Error
sqrt(mean((pred-test.label)^2))
```

```
[1] 1.056506
```

Generate and Save Histogram of Error

```
error = pred-test.label
DOYWeathPrev = data.frame(actual = test.label, predicted=pred, err = error)

errplt = ggplot(DOYWeathPrev, aes(x=err)) +
 geom_histogram(aes(y=..density..), colour="black", fill="grey")+
 xlim(-6.5,6.5) +
 ggtitle("Model 4 Error") +
 xlab("Error (\u00B0C)") +
 ylab("Density")  + theme(text = element_text(size = 30))

ggsave(plot = errplt, width = 7.5, height = 4.5, dpi = 340, filename = "DOYWeathPrev7Error.PNG")
```

Generate and Save Predicted vs. Actual Plot

```
abplt = ggplot(DOYWeathPrev, aes(x = actual, y = predicted)) + geom_point(shape=23)+ ggtitle("Model 4 Predicti
 xlab("Predicted Value (\u00B0C)") +
 ylab("Actual Value (\u00B0C)") + geom_abline() + theme(text = element_text(size = 30))


ggsave(plot = abplt, width = 7.5, height = 4.5, dpi = 340, filename = "DOYWeathPrev7plot.PNG")
```