"Predictive Indices from the Susquehanna River for ChlA Concentration at the Head of the Chesapeake Bay"

Britton Hartzok

4/30/2021

Introduction

The Susquehanna River alone introduces 40% of fresh water to the Chesapeake Bay which majorly impacts the delicate chemistry of this vast ecosystem. The purpose of my analysis is to create a predictive model for chlorophyll-a concentrations at the head of the Bay by using water quality metrics of the lower and middle portions of the Susquehanna River. My data has been sourced from two authorities - the USGS Stream Monitoring Network as well as the Chesapeake Bay Program's Nontidal Water Quality Monitoring (NTWQM) and Tidal Water Quality Monitoring (TWQM) programs. I only examined 4 years of data (2016-2020) which greatly limited my choices of machine learning algorithms. While I created some relatively successful models, my accuracy and generalized fit would've been made more credible had I studied a 10 or 20 year span.

The upper Chesapeake Bay is famous for Spring diatom blooms. However, because I didn't examine one particular type of phytoplankton, I really had no threshold measurement for what would correspond to a bloom for any given species. This limited my approach to using only numerical predictions, although I'd be very interested in working with classifiers in the future. With the exception of continuous in-situ WQ monitoring stations, most of my variables were only measured on a monthly or biweekly basis. With sparse data for each day or week, I chose to create monthly averages which further constricted my analysis (n=48). However, as you will see, my results were rather satisfactory and are useful in the broader context of understanding the interconnected relationship of the Susquehanna and Chesapeake systems.

Methodology

I started out this analysis with 109 predictor variables. After much trimming and regressing, I was able to hone my dataset down to just 30 of the most significant predictor features. After noticing my models were alarmingly accurate, I became suspicious. Upon further investigation, I found two predictor variables needed to be eliminated for my analysis to be valid. "BelowDamChla" clearly has some collinearity to downstream chlorophyll-a at station CB1.1, and "CBPHEO" (pheophytin) is a metric for calculating the Phytoplankton Index of Biotic Integrity (PIBI). Both of these variables compromise the coherence of my model and were removed. This change greatly reduced the accuracy of all my models. However, as you'll see, select machine learning methods were pretty resilient to the change - a testement to their power and usefulness.

Variable Selection (48 obs. of 31 variables)

(**Locations Listed Geographically North-South**) **Water Temperature (°C):**
- DanvilleTempC (Susquehanna River - Danville)

- JuniataT (Juniata River - Newport, Perry County)
- PaxtonT (Paxton Creek - Dauphin County)
- CityIslandT (City Island-Susquehanna River - Harrisburg)
- BreechesT (Yellow Breeches - Cumberland County)
- eConewagoT (Conewago Creek - Lancaster County)
- wConewagoT (Conewago Creek - York County)
- ConestogaT (Conestoga River - Lancaster County)
- MariettaT (Chiques Creek/Susquehanna - Lancaster County)
- MuddyCreekT (Muddy Creek - York County)
- DamT (Conowingo Dam - Maryland)
- BelowDamTemp (Below Conowingo Dam - Maryland)
- CBWTEMP (CB Mainstem Station 1.1 - Maryland)

Dissolve Oxygen (mg/L) - JuniataDO - BreechesDO - wConewagoDO - MariettaDO - ConestogaDO - MuddyRunDO
- BelowDamDO
- DamDO
- CBDO

Nitrogen (mg/L)
- ConestogaN (Total N)
- CBPN (Particulate N)

Phosphorus (mg/L)
- JuniataP (Total P)
- CBTP (Total P)

Silica (mg/L)
- DamSIF (Filtered Si- perhaps related to diatoms)

Carbon (mg/L) - CBPC (Particulate C)

pH (standard units)
- CBPH

Sigma T (No Units) - CBSIGMA_T (Water Density as Function of (WTEMP, Salinity)

Chlorophyll-a (ug/L) - BayChla (Our dependent variable)

Analysis

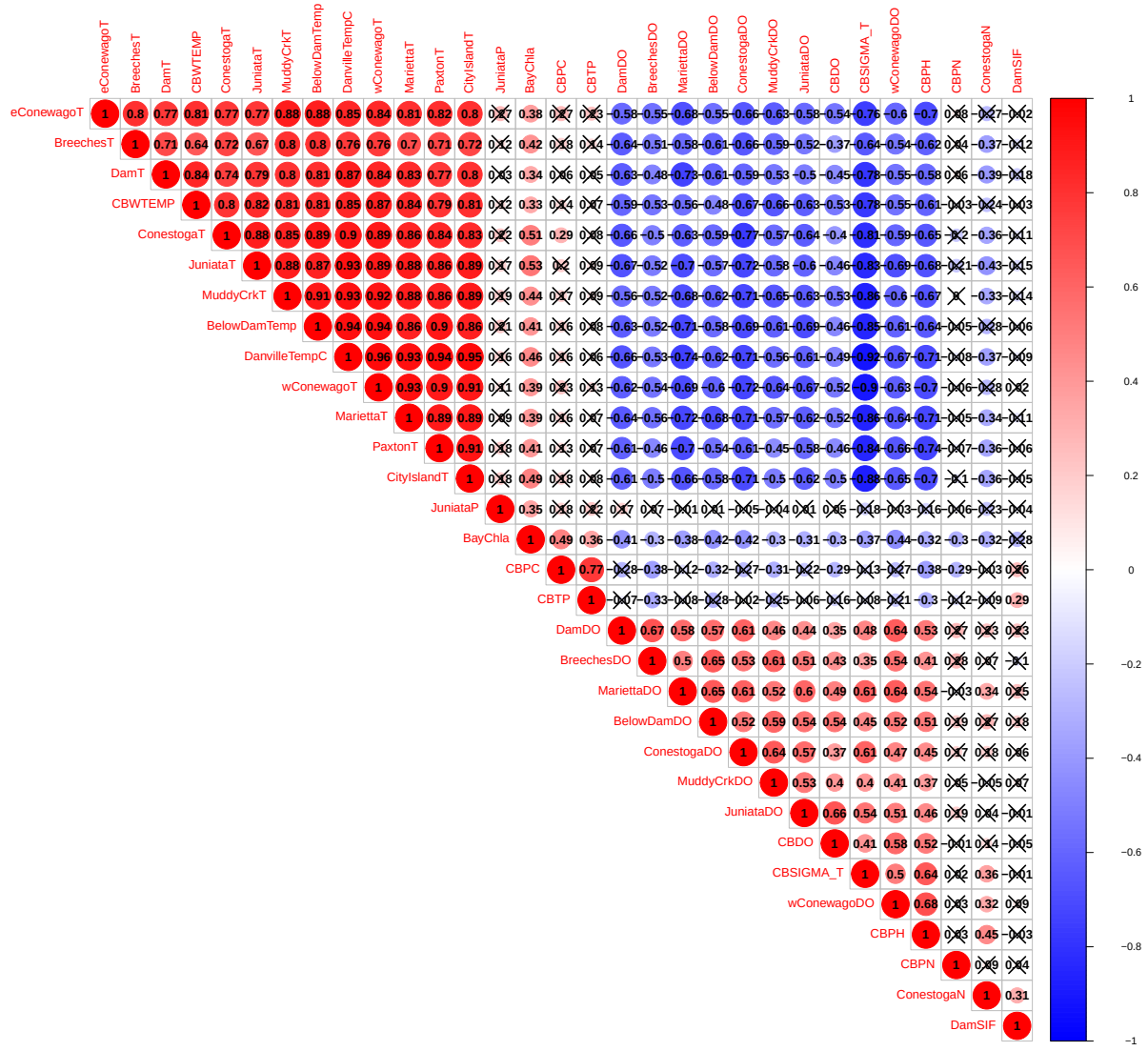
Data Read-In

Missing values will be imputed using Multiple Imputation by Chained Equations

```
knitr::opts_chunk$set(fig.width=16, fig.height=16)
wq<- read.csv("algae.csv", header=T)
#MICE
wq <- mice(wq,m=5,maxit=10,meth='pmm',seed=320)
# Completed imputation, selecting our completed dataframe
wq <- complete(wq,1)
# Check for NAs - Returns 'FALSE', No N/As
anyNA(wq)
# Summary Statistics
describe(wq)
```

Scaling, Centering, Basic Correlation Between Variables

```
preProcValues <- preProcess(wq, method = c("center", "scale"))
numvars <- predict(preProcValues, wq)
res2 <- rcorr(as.matrix(numvars))
# Insignificant correlations, at alpha=0.05, are crossed out
set.seed(31)
corrplot(res2$r, type="upper", order="hclust", addCoef.col = TRUE,
         col=colorRampPalette(c("blue","white","red"))(200),
         p.mat = res2$P, sig.level = 0.05, insig = "pch")
```



Multivariate Linear Regression

```
knitr::opts_chunk$set(fig.width=6, fig.height=6)
# Multivariate Linear Regression
set.seed(412)
lm.all <- lm(BayChla~., data=wq)
wq<-wq[,-c(6,9,12,15,16,22)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(1,4,21,22)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(1,8,19)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(8)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(8,11)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(4,7,11)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(1,2,11)]
lm.tune <-lm(BayChla~., data=wq)
wq<-wq[,-c(1,4)]
lm.final <-lm(BayChla~., data=wq)
summary(lm.final)
```

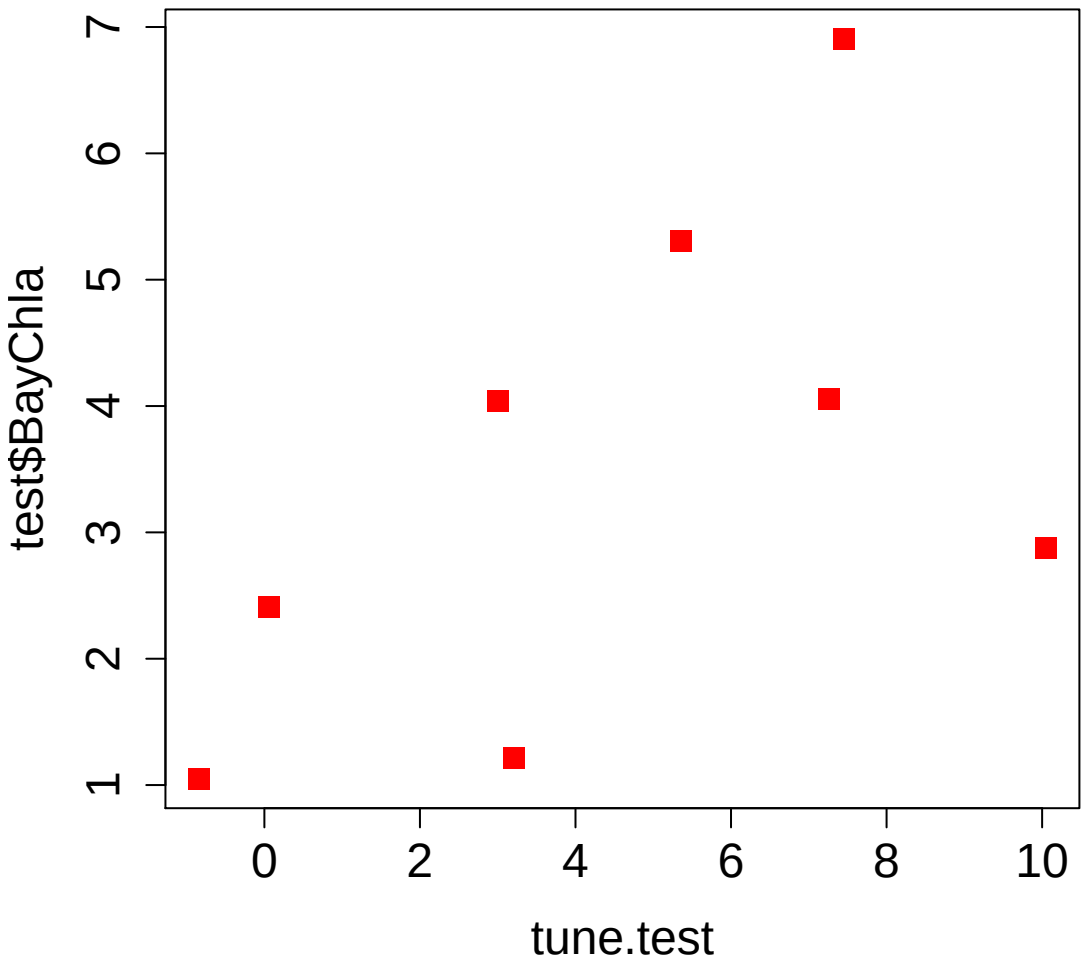
```
##
## Call:
## lm(formula = BayChla ~ ., data = wq)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.824 -1.149  0.162  1.014  7.053
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -23.55250    9.47090  -2.487 0.017051 *
## JuniataT      0.23169    0.06026   3.845 0.000413 ***
## MuddyCrkDO    0.20688    0.09314   2.221 0.031916 *
## DamSIF       -1.82534    0.56563  -3.227 0.002461 **
## BreechesT     0.12235    0.06656   1.838 0.073289 .
## CBPC          5.87985    1.03868   5.661 1.31e-06 ***
## CBPH          2.44028    0.98200   2.485 0.017126 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.995 on 41 degrees of freedom
## Multiple R-squared:  0.627,  Adjusted R-squared:  0.5724
## F-statistic: 11.49 on 6 and 41 DF,  p-value: 1.668e-07
```

Now that certain features have been eliminated, let's try linear regression on a training and test set.

```r
set.seed(1981)
#Training/test set at 80:10 ratio
trainIndex <- createDataPartition(wq$BayChla,
                                  p = 0.8, list = FALSE, times = 1)
#Parse out training/test sets
train <- wq[ trainIndex,] #n=40
test <- wq[-trainIndex,] #n=8

set.seed(412)
lm.all <- lm(BayChla~., data=train)
train<-train[,-c(4)]
final.tune <-lm(BayChla~., data=train)
summary(final.tune)
```

```
##
## Call:
## lm(formula = BayChla ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2842 -1.1719  0.1754  1.0103  6.1879
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -23.90561    9.16000  -2.610 0.013375 *
## JuniataT      0.34636    0.06611   5.239 8.42e-06 ***
## MuddyCrkDO    0.19779    0.08865   2.231 0.032382 *
## DamSIF       -2.31585    0.61033  -3.794 0.000581 ***
## CBPC          6.61744    1.04962   6.305 3.47e-07 ***
## CBPH          2.56191    0.98128   2.611 0.013343 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.861 on 34 degrees of freedom
## Multiple R-squared:  0.7123, Adjusted R-squared:   0.67
## F-statistic: 16.84 on 5 and 34 DF,  p-value: 2.307e-08
```

```r
tune.test<-predict(final.tune,newdata=test)
plot(tune.test,test$BayChla,cex=1.5, cex.lab=1.5, cex.axis=1.5, col="red", type="p", pch=15)
```




```
tune.test.resid<-tune.test-test$BayChla

#Mean Absolute Error (MAE)
linregMAE<-sum(abs(tune.test.resid))/nrow(test)
linregMAE
```

```
## [1] 2.280986
```

```
#Root Mean Squared Error
linregRMSE<-sqrt(mean(tune.test.resid^2))
linregRMSE
```

```
## [1] 3.084639
```

With a prediction accuracy of MAE= $\pm$ 2.28 mg/L Chla and a RMSE = 3.08, this linear regression model isn't extremely accurate. Next, I'm going to feed the entire untrimmed dataset into a support vector machine (SVM) with both linear and radial kernals as well as an elastic net regression fit model.

Machine Learning Techniques for Variable Prediction (SVMs and Elastic Net)

Cost-Tuned Linear SVM on Full Data Set

```
# Train control with repeated 10-fold cross validation, repeated 3 times
# Need to be careful with trctrl function to avoid "overtuning",
## R-Squared=1 isn't always a good thing
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
# Training method, preprocessing
svm_Linear <- train(BayChla ~., data = train, method ='svmLinear',
                    trControl=trctrl, tuneLength = 10,
                    preProcess = c("center", "scale"))
svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 40 samples
## 30 predictors
##
## Pre-processing: centered (30), scaled (30)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 36, 35, 37, 37, 36, 36, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   3.664094  0.3015589  2.995365
##
## Tuning parameter 'C' was held constant at a value of 1
```

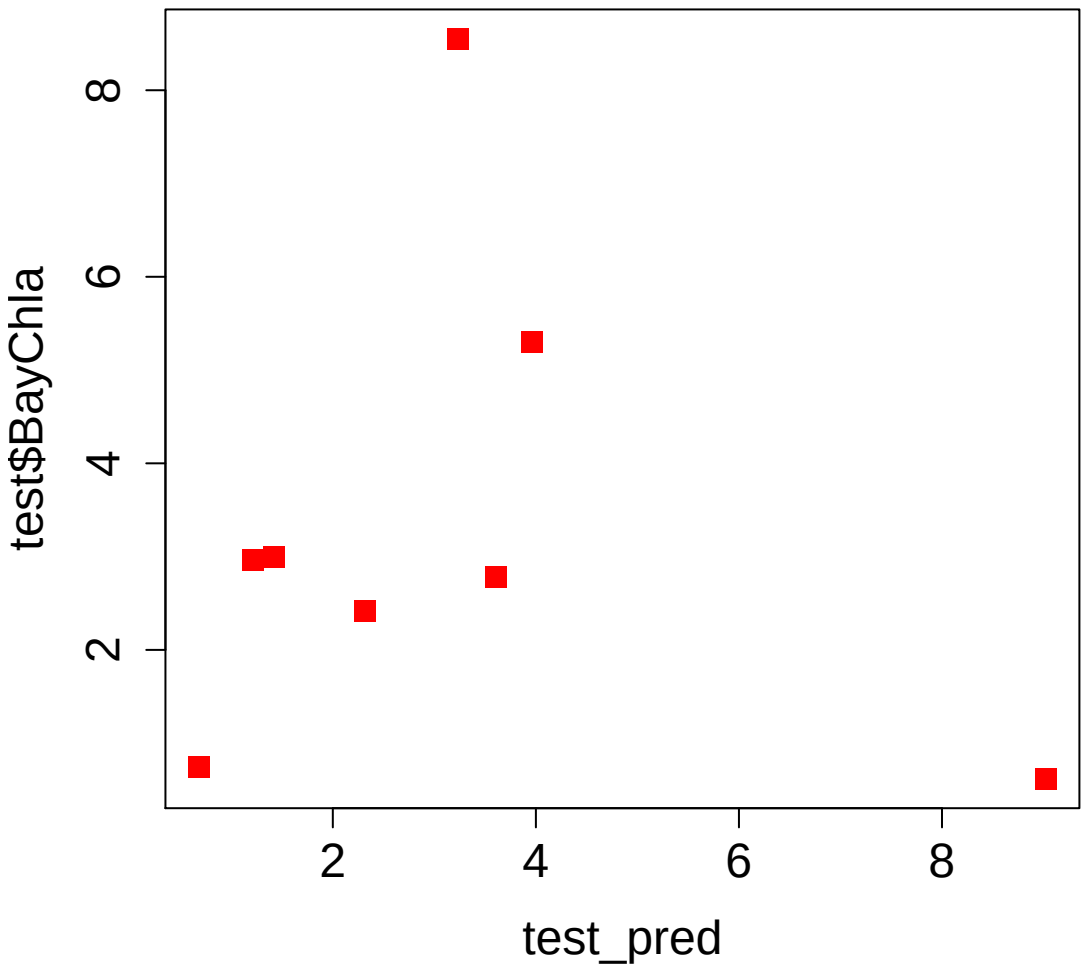
```
test_pred <- predict(svm_Linear, newdata = test)
resids<-test_pred-test$BayChla

#Mean Absolute Error (MAE)
linsvmMAE<-sum(abs(resids))/nrow(test)
linsvmMAE
```

```
## [1] 2.422026
```

```
#Root Mean Squared Error
linsvmRMSE<-sqrt(mean(resids^2))
linsvmRMSE
```

```
## [1] 3.654987
```

```
plot(test_pred,test$BayChla, cex=1.5, cex.lab=1.5, cex.axis=1.5, col="red", type="p", pch=15)
```



Sigma/Cost-Tuned Radial SVM Tuning on Original Training Set

```
svm_rbf <- train(BayChla ~., data = train, method ='svmRadial',
                 trControl=trctrl, tuneLength = 10,
                 preProcess = c("center", "scale"))
svm_rbf
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 40 samples
## 30 predictors
##
## Pre-processing: centered (30), scaled (30)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 37, 35, 37, 36, 36, 36, ...
## Resampling results across tuning parameters:
##
```

```
##   C       RMSE      Rsquared   MAE
##     0.25  2.378375  0.5729166  1.839877
##     0.50  2.290754  0.5597836  1.795598
##     1.00  2.237112  0.5519931  1.769375
##     2.00  2.180154  0.5344902  1.756969
##     4.00  2.187094  0.5292108  1.750400
##     8.00  2.302807  0.5167578  1.853620
##    16.00  2.492595  0.5095429  1.999945
##    32.00  2.492595  0.5095429  1.999945
##    64.00  2.492595  0.5095429  1.999945
##   128.00  2.492595  0.5095429  1.999945
##
## Tuning parameter 'sigma' was held constant at a value of 0.0245123
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.0245123 and C = 2.
```

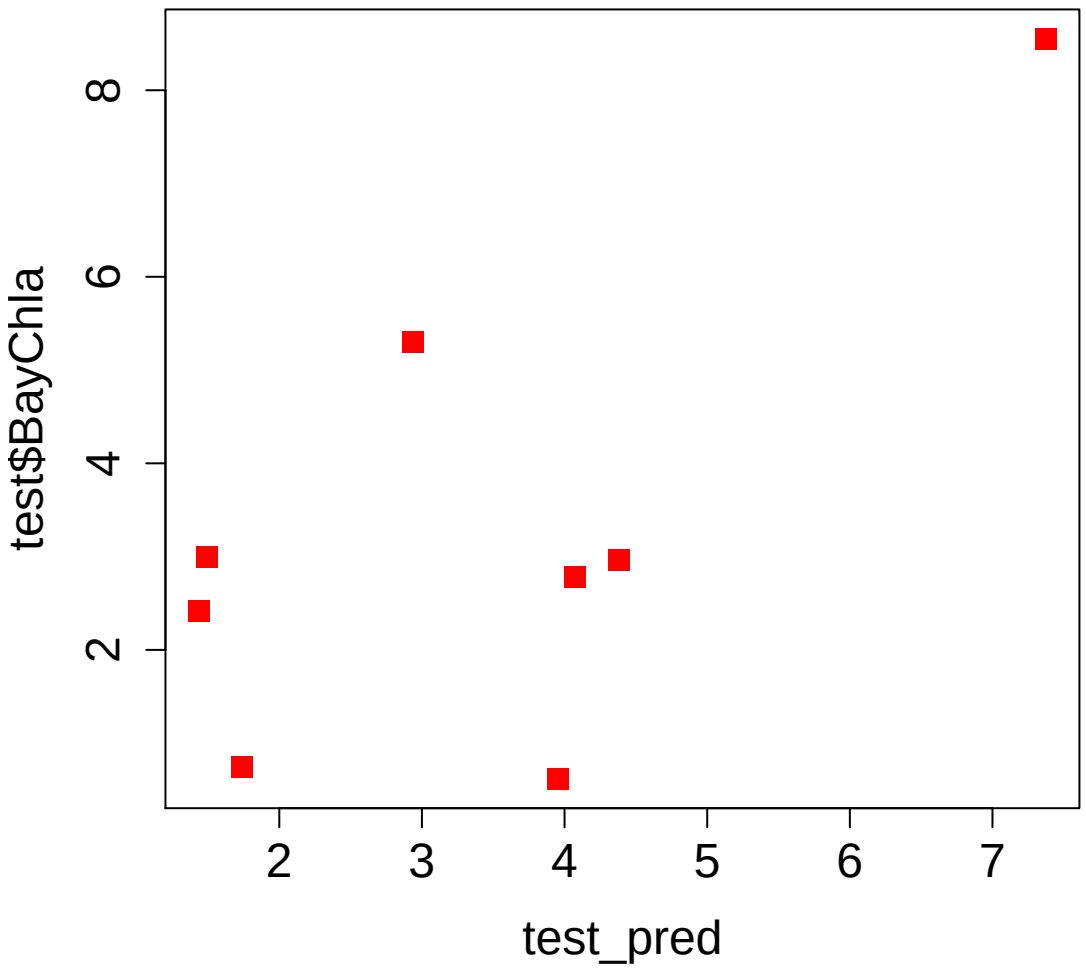
```
test_pred <- predict(svm_rbf, newdata = test)
rbf.resids<-test_pred-test$BayChla

#Mean Absolute Error (MAE)
rbfMAE<-sum(abs(rbf.resids))/nrow(test)
rbfMAE
```

```
## [1] 1.632891
```

```
#Root Mean Squared Error
rbfRMSE<-sqrt(mean(rbf.resids^2))
rbfRMSE
```

```
## [1] 1.80266
```

```
plot(test_pred,test$BayChla,cex=1.5, cex.lab=1.5, cex.axis=1.5, col="red", type="p", pch=15)
```



Elastic Net Learning Algorithm Tuned with Fraction of Full Solution (FFS) and Weight of Decay

```
## Elastic Net Tuned on Fraction of Full Solution and Weight Decay
enet <- train(BayChla ~., data = train, method ='enet',
              trControl=trctrl, tuneLength = 10,
              preProcess = c("center", "scale"))
#Best Fraction and Lambda (Decay) coefficients
enet$bestTune
```

```
##     fraction lambda
## 93 0.2611111    0.1
```

```
test_pred <- predict(enet, newdata = test)
enet.resids<-test_pred-test$BayChla

#Mean Absolute Error (MAE)
```

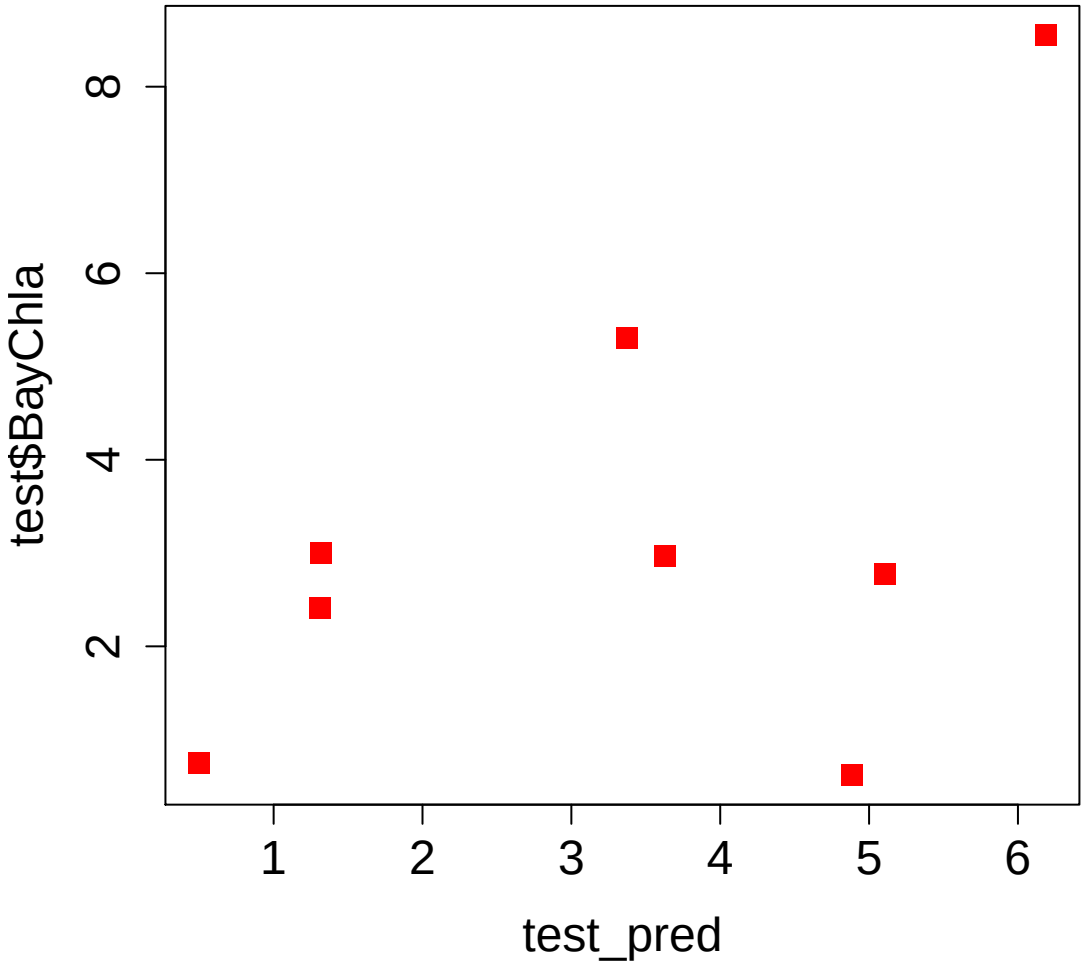
```r
enetMAE<-sum(abs(enet.resids))/nrow(test)
enetMAE
```

```
## [1] 1.822627
```

```r
#Root Mean Squared Error
enetRMSE<-sqrt(mean(enet.resids^2))
enetRMSE
```

```
## [1] 2.163941
```

```r
plot(test_pred,test$BayChla,cex=1.5, cex.lab=1.5, cex.axis=1.5, col="red", type="p", pch=15)
```



	Linear Regression				Linear SVM				Radial SVM				Elastic Net			
Out of	RSE	R2	Adj.R2	P-Val	RMSE	R2	Cost	CV	RMSE	R2	MAE	Cost	RMSE	R2	MAE	Lambda
Box Stats	1.861	0.71	0.67	0	3.66	0.3	1	10-Fold	2.18	0.53	1.76	2	2.02	0.62	1.67	0.1
Test Set	MAE		RMSE		MAE		RMSE		MAE		RMSE		MAE		RMSE	
Accuracy	2.28		3.08		2.4		3.66		1.63		1.8		1.82		2.2	

Figure 1: Summary Table

Discussion

Analysis was very limited by my low number of observations (n=48). I operated on a small training set (n=40) and predicted on an even smaller test set (n=8). In the future, I'd like to investigate 10-20 years worth of data rather than 4. With the exception of the linear SVM, I was mostly pleased with the improvements seen with the various machine learn methods. The challenge with opting for higher power models is that you lose flexibility and often overfit the model to your particular set of data rather than creating a model that can generalize and remain accurate on new sets of data. With that being said, I tried to keep my tuning techniques to a minimum so as to simplify model explanation and increase ease of replication.

Although the linear regression model has the highest R^2 value, the root-mean-squared-error(RMSE) observed on the test set disqualified this model for me. The linear SVM performed notably worse with a RMSE of 3.66. The Radial SVM and the Elastic Net were the strongest performers. At a mean-absolute-error(MAE) of between ±1.63-1.82 mg/L ChlA, they are both within a tolerable range of error. In addition, their respective RMSEs on the test set are dramatically lower than the linear regression.

I tried to approach this data from a classification perspective as well. I 'binned' the ChlA concentrations on a set scale of 0.5 and 1 mg/L increments, however this obviously severely limited class counts and proportions. In some cases, the training module only had 1 opportunity to learn any one class. Likewise, my test set often had just one instance of each class which lowered accuracy dramatically. Therefore, if I were to go back and retrieve a greater time span of data, I would also attempt classification methods in my machine learning process.