Project Report

Zack Laird

2023-12-05

Introduction

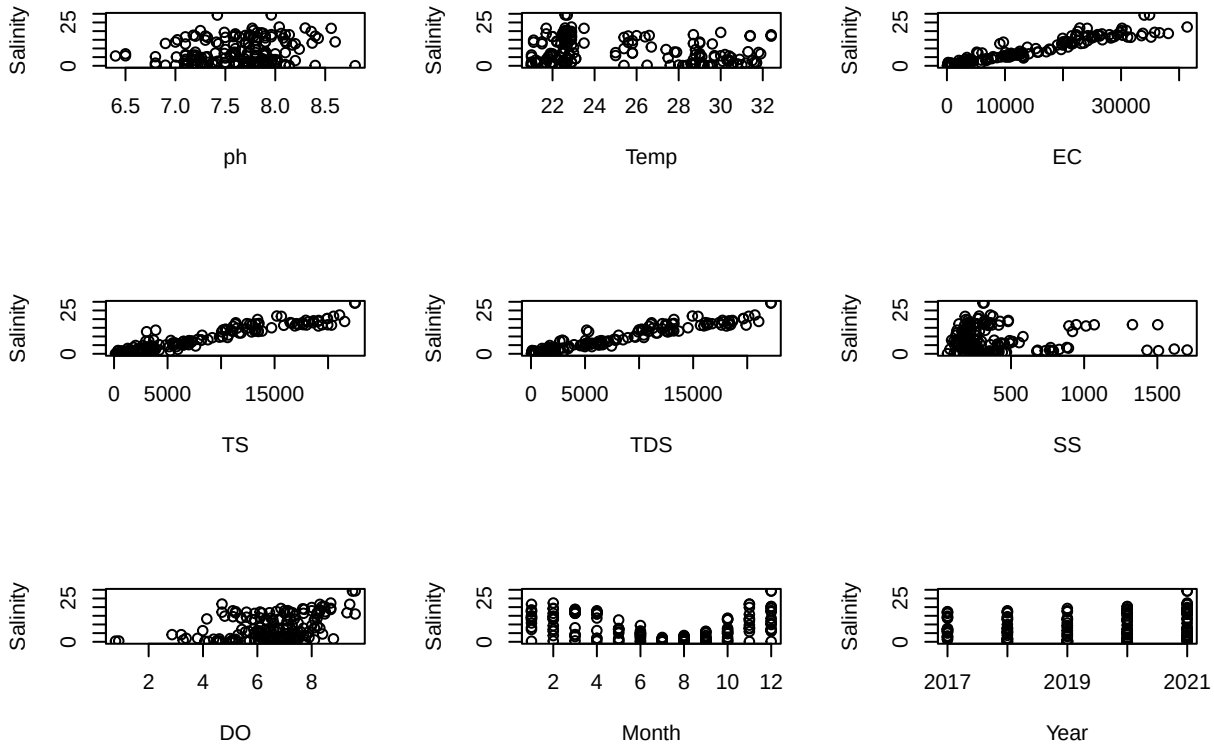
The Karnaphuli River in Bangladesh is a vital resource to those living in the region. It provides transportation for local travel and trade, hydroelectric power for industries, and a reservoir for agricultural usage. Because the river is so important to the region, collection of data on the river is necessary to make predictions and inference about the quality of the water. The data set used for this project contains 171 observations and 11 variables which are shown below. This project will focus on making predictions based on `Salinity` and `Location`. The goal of this project is to build machine learning models capable of predicting the level of salinity in a sample both based on a high/low criteria and as general levels, and to create a model that can predict the locations from which samples were drawn.

```
##   Year Location Temp  ph    EC    TS   TDS  SS  DO Salinity Month
## 1 2017     CUFL 22.2 8.1 20450 10414 10228 186 5.6     14.2     1
## 2 2017      TSP 22.2 7.9 19984 10190  9992 198 6.1     13.8     1
## 3 2017     CUFL 21.1 7.3 22580 11494 11292 202 4.8     14.8     2
## 4 2017      TSP 21.1 7.9 21356 10906 10678 228 5.4     13.6     2
## 5 2017      TSP 25.6 7.3 26330 13476 13168 308 6.1     17.2     3
```

Methodology

Initial Exploration

I began by creating scatterplots of potentially useful predictors against `Salinity`. `TS`, `TDS`, and `EC` (total solids, total dissolved solids, and electrical conductivity respectively) jumped out at me as the most useful given their strong linear relationships to the target variable.



LDA and QDA Models

My first goal was to build a model to classify salinity levels by a high/low type. I chose to make the high/low criteria based on whether an observation had a salinity level above the mean the whole salinity vector. If the level is above the mean it is denoted by 1, else 0. I created the dummy variable and after splitting the data into training and testing sets (roughly 70/30 split) both LDA and QDA models were fit to the training set. As shown in the confusion matrices for LDA and QDA respectively, both models performed equally well. Both had test error rates of only 3.9%, so both are potentially useful models for use as binary classifiers.

```
# Create dummy var
Salinity.type <- rep(0, dim(water)[1])
Salinity.type[Salinity > mean(Salinity)] = 1
water$Salinity.type <- Salinity.type


# Split data
set.seed(123)
train <- sort(sample(1:dim(water)[1], 120))
water.train <- water[train, ]
water.test <- water[-train, ]

# LDA
water.lda <- lda(Salinity.type ~ EC + TS + TDS, data = water.train)
pred.table <- table(predict(water.lda, water.test)$class, water.test$Salinity.type)
pred.table
```

```
##
##      0  1
##   0 31  2
##   1  0 18
```

```
(pred.table[1,2] + pred.table[2,1]) / dim(water.test)[1]
```

```
## [1] 0.03921569
```

```
# QDA
water.qda <- qda(Salinity.type ~ EC + TS + TDS, data = water.train)
pred.table <- table(predict(water.qda, water.test)$class, water.test$Salinity.type)
pred.table
```

```
##
##      0  1
##   0 30  1
##   1  1 19
```

```
(pred.table[1,2] + pred.table[2,1]) / dim(water.test)[1]
```

```
## [1] 0.03921569
```

Multiple Linear Regression

Next, I fit a multiple linear regression model using the predictors identified previously as having strong linear relationships with the target variable. Since `EC` had a p-value less than $\alpha = 0.05$ in the first model, it was dropped as a predictor. The final model is Salinity = 0.95 - 0.001 * TS + 0.002 * TDS.
All predictors are now statistically significant. The calculated test MSE for the model was 3.05, in the context of the range of the target variable being from 0.05 - 29.34 this is a good MSE. To illustrate the fit of the model, I created a 3D scatterplot of the data including a plane to show the models fit. The model had an RSq of 92.45%, meaning that the model is a very good fit for the data and explains 92.45% of the variance.

```
# Multiple Linear Regression
water.mlr <- lm(Salinity ~ EC + TS + TDS, data = water.train)
summary(water.mlr)$coef # Drop EC, not significant
```

```
##                  Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)  0.8556430307 0.3297705945  2.594661 0.0106899863
## EC           0.0001100199 0.0001033390  1.064650 0.2892454128
## TS          -0.0011310175 0.0004462283 -2.534616 0.0125893399
## TDS          0.0020159327 0.0005069996  3.976202 0.0001221913
```

```
water.mlr <- lm(Salinity ~ TS + TDS, data = water.train)
summary(water.mlr)$coef
```

```
##                 Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)  0.949240969 0.3180172553  2.984873 3.454953e-03
## TS          -0.001208223 0.0004405475 -2.742549 7.055575e-03
## TDS          0.002279319 0.0004427968  5.147551 1.077459e-06
```

```
summary(water.mlr)$r.squared
```

```
## [1] 0.924484
```

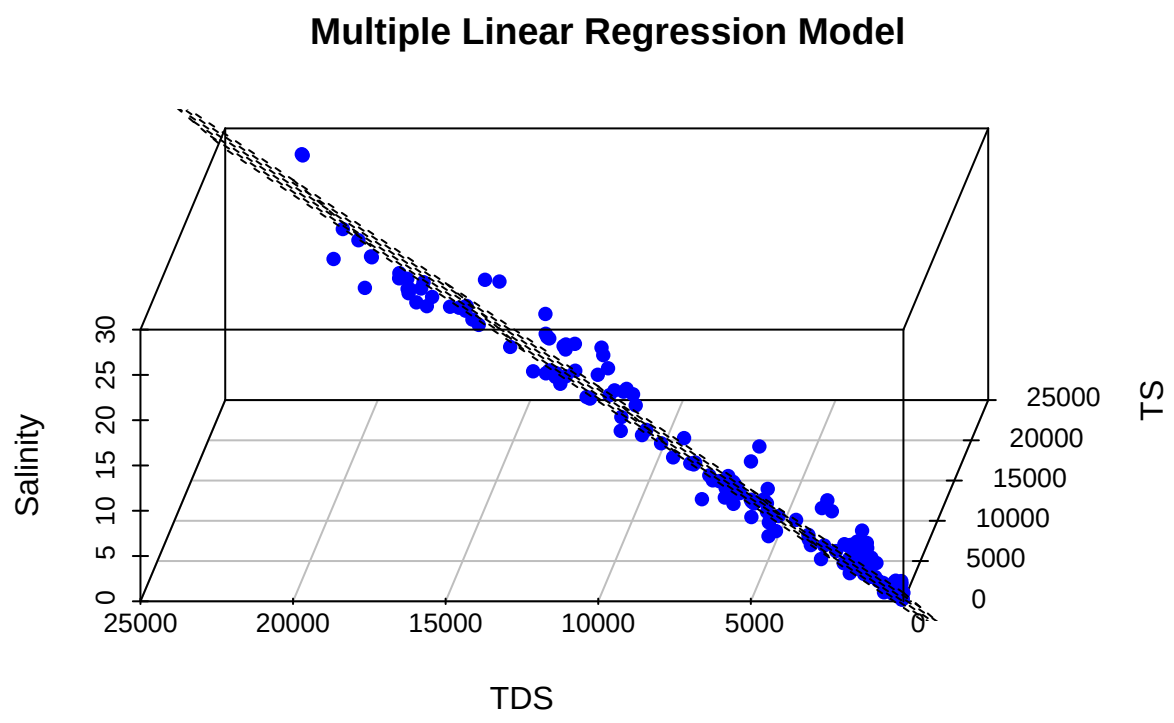
```
water.preds <- predict(water.mlr, water.test)
MSE <- mean((water.preds - water.test$Salinity)^2)
MSE
```

```
## [1] 3.054514
```

```
range(Salinity)
```

```
## [1]  0.05 29.34
```

```
# Plot and show how the model fit
par(mfrow=c(1,1))
s3d <- scatterplot3d(TS, TDS, Salinity, color = "blue", pch = 16,
                     main = "Multiple Linear Regression Model",
                     angle = 260)
s3d$plane3d(water.mlr)
```



KNN Classification

The next point of focus for this project is building a classifier model for the `Location` variable. I started by creating a KNN classifier using the `TS` and `TDS` variables. I used a `for` loop to determine the optimal choice for k. I tested k values between 1 and 50 and stored the accuracy score for each model to a vector. The optimal value for k was 8, and this model was accurate about 55% of the time. Since the error was relatively low I did some checking into why this could be. First, compared to the expected baseline accuracy of 25% (which would be the case if we randomly guessed each class) the accuracy of this model is decently high. Secondly, I checked the class distribution and found that the classes were disproportionately distributed which could have caused a higher error rate.

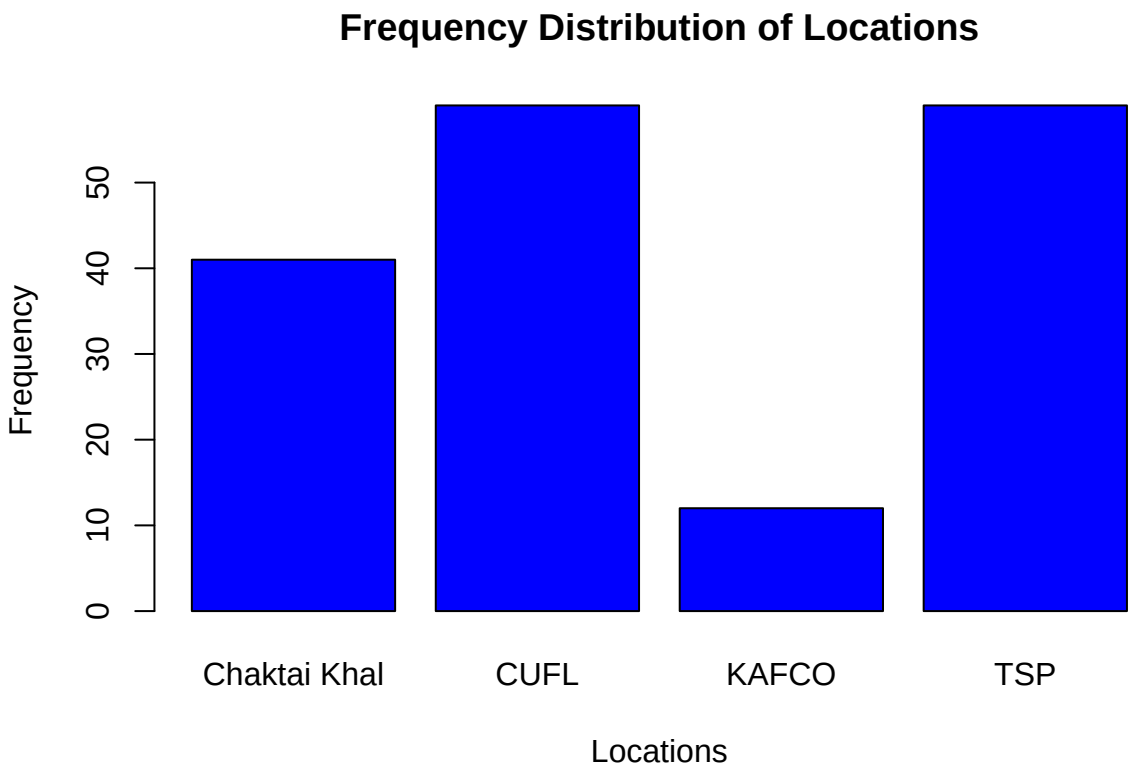
```
# get the index of the min in the list to determine best choice k
errVec <- rep(0, 50)
for(i in 1:50){
  knn.preds <- knn(train = water.train[,6:7], test = water.test[,6:7],
                   cl = water.train$Location, k = i)
  count <- rep(0, 51)
  for(j in 1:51){
    if(knn.preds[j] == water.test$Location[j]){
      count[j] <- 1
    }
}
  errVec[i] <- 1 - sum(count)/length(count)
}

which.min(errVec) # optimal k is k = 8
```

```
## [1] 8
```

```
min(errVec) # model was wrong 45.1% of the time
```

```
## [1] 0.4509804
```

```
plot(as.factor(Location), xlab = "Locations", ylab = "Frequency",
     main = "Frequency Distribution of Locations", col = "blue")
```



Decision Trees

Following KNN classification I tried decision tree models to see if they had improved performance over KNN. I fit the first tree using all the predictors, which netted an accuracy score of about 51%, slightly worse than KNN. Pruning the tree using cross validation improved the models performance slightly bumping the accuracy score up to 55%, interestingly enough performing exactly how well KNN performed.

```
### Classification Tree
set.seed(1126)
water$Location <- as.factor(water$Location)
tree.water <- tree(Location ~ ., subset = train, data = water)
plot(tree.water)
text(tree.water, pretty = 0)

# Predictions on the test set
location.test <- water$Location[-train]
location.pred <- predict(tree.water, newdata = water.test, type = "class")
table(location.pred, location.test)
```

```
##                location.test
## location.pred   Chaktai Khal CUFL KAFCO TSP
##    Chaktai Khal            8    1     1   3
##    CUFL                    0    3     0   1
##    KAFCO                   0    3     0   1
##    TSP                     0   12     3  15
```

```
(8+3+15)/nrow(water.test)
```

```
## [1] 0.5098039
```

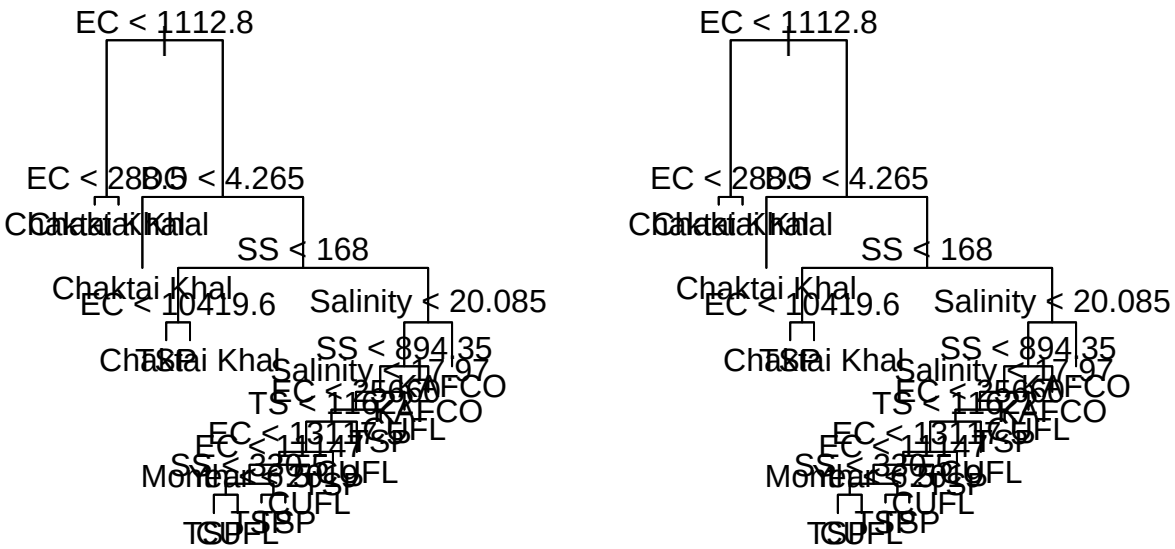
```
# Cross Validate
cv.water <- cv.tree(tree.water, FUN = prune.misclass)
which.min(cv.water$dev)
```

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

```
cv.water$size[1]
```

```
## [1] 16
```

```
prune.water <- prune.misclass(tree.water, best = 16)
par(mfrow = c(1,2))
plot(tree.water)
text(tree.water, pretty = 0)
plot(prune.water)
text(prune.water, pretty = 0)
```



```
# Preds using pruned tree
location.pred <- predict(prune.water, newdata = water.test, type = "class")
table(location.pred, location.test)
```

```
##                location.test
## location.pred   Chaktai Khal CUFL KAFCO TSP
##    Chaktai Khal            8    1     1   3
##    CUFL                    0    3     0   1
##    KAFCO                   0    3     1   0
##    TSP                     0   12     2  16
```

```
(8+3+1+16)/nrow(water.test)
```

```
## [1] 0.5490196
```

Results

In summary, for binary classification of salinity levels LDA and QDA models performed equally well, both giving very low error rates at only 3.9% misclassification. The multiple linear regression model also performed exceptionally well and had a very low mean squared error in the context of the range of the target variable. Additionally, this model had an RSq value of 92.45%, meaning that the model was a very good fit for the data and explains 92.45% of the variance. The KNN classifier was reasonably successful in classifying the locations that the samples were drawn from given 4 classes were being predicted. The decision tree model performed slightly worse than the KNN model, but after pruning the decision tree using cross validation, it was found that the cross validated tree performed equally as well as the KNN model.

Discussion

For future work I will likely focus on finding an optimal set of predictors for both KNN classification and decision trees. I could also potentially find a way to normalize the distribution of the locations, as these were quite skewed. Further, trying different models such as random forest or SVM could potentially have improved accuracy over KNN and decision trees.