Disaster Relief Project: Part II Model Build & Test Data, apply to Holdout

Ashlie Ossege ajo5fs

May 10, 2021

1 Introduction

It was 2012. I was floating on a raft in the crystal clear Haitian waters with the sun warming my face, and coconut and fruit on my tongue. Peering though my Ralph Lauren sunglasses, I catch sight of a deflated plastic juice-bag-shaped object floating by. I could make out the word 'water'. In that moment I was hit with the bleak reminder of the devastation that occurred just on the other side of the beautiful mountain tops in 2010. The earthquake that left families and children hungry, longing for food and shelter and suffering the loss of loved ones. These water sachets or 'saches dlo' are commonly distributed throughout Haiti in times of disaster and times of peace, providing potable water to those who call Haiti their home.

How was aid distributed to the 300,000 people that were injured in the earthquake and 5 million* displaced? Analysis of the disaster relief data reveals the power of data mining in helping provide actionable insights for social aid for incomprehensible conditions.

*Worldvision.org

2 Training Data / EDA

```
# Load Required Packages
library(tidyverse)
library(broom)
library(yardstick)  ##https://yardstick.tidymodels.org/reference/roc_auc.html
library(kableExtra)
library(ISLR)
library(pheatmap)
library(dplyr)
library(ggplot2)
library(ROCR)
library(MASS)
library(class)
library(glmnet)
library(knitr)


setwd("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021")
haiti <- read.csv(file = 'HaitiPixels.csv')
head(haiti)
```

```
##        Class Red Green Blue
## 1 Vegetation  64    67   50
## 2 Vegetation  64    67   50
## 3 Vegetation  64    66   49
## 4 Vegetation  75    82   53
## 5 Vegetation  74    82   54
## 6 Vegetation  72    76   52
```

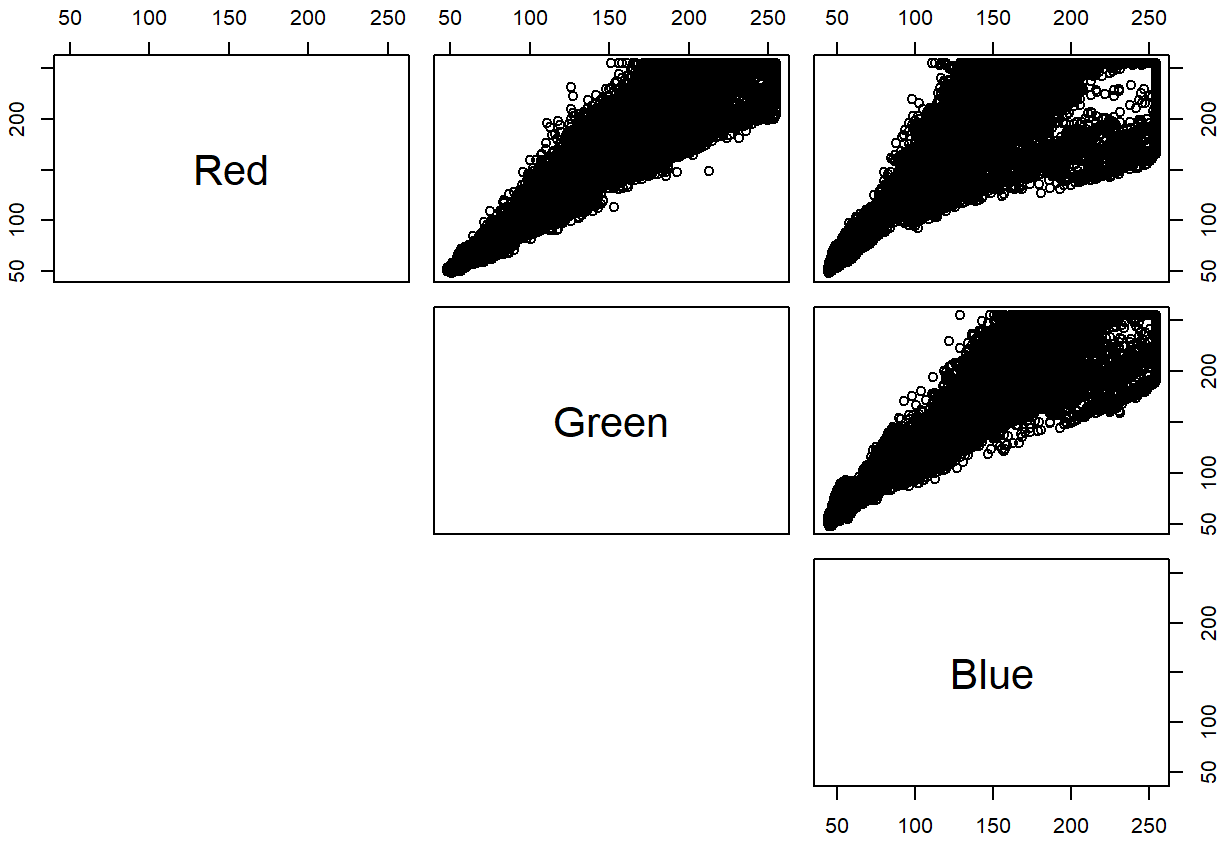
```
summary(haiti)
```

```
##     Class                Red          Green            Blue
##  Length:63241       Min.   : 48   Min.   : 48.0   Min.   : 44.0
##  Class :character   1st Qu.: 80   1st Qu.: 78.0   1st Qu.: 63.0
##  Mode  :character   Median :163   Median :148.0   Median :123.0
##                     Mean   :163   Mean   :153.7   Mean   :125.1
##                     3rd Qu.:255   3rd Qu.:226.0   3rd Qu.:181.0
##                     Max.   :255   Max.   :255.0   Max.   :255.0
```

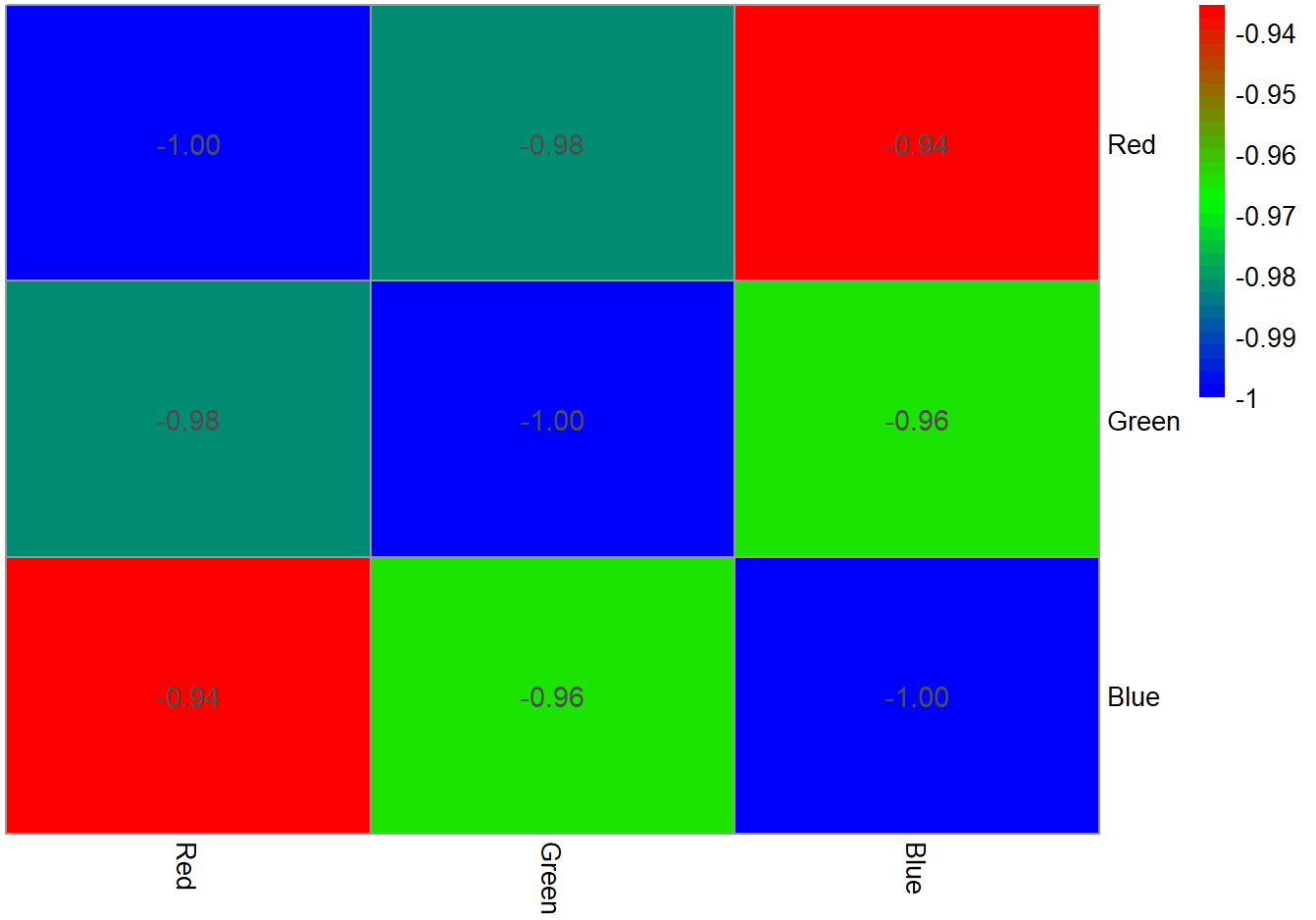
```
attach(haiti)
mytable <-table(Class)
mytable
```

```
## Class
##        Blue Tarp          Rooftop             Soil Various Non-Tarp
##             2022             9903            20566             4744
##       Vegetation
##            26006
```

```
pairs(haiti[sapply(haiti,is.numeric)], lower.panel = NULL)
```




```
library(pheatmap)
haiti.corr<--cor(haiti[sapply(haiti,is.numeric)])
pheatmap(haiti.corr, display_numbers = T, color = colorRampPalette(c("blue", "green", "re
         d"))(30), cluster_rows = F, cluster_cols = F, fontsize_number = 11)
```




```
library(dplyr)
haiti %>%
  group_by(Class) %>%
  summarize(COR=cor(Blue,Red))
```

```
## # A tibble: 5 x 2
##   Class              COR
## * <chr>            <dbl>
## 1 Blue Tarp        0.859
## 2 Rooftop          0.946
## 3 Soil             0.660
## 4 Various Non-Tarp 0.898
## 5 Vegetation       0.946
```

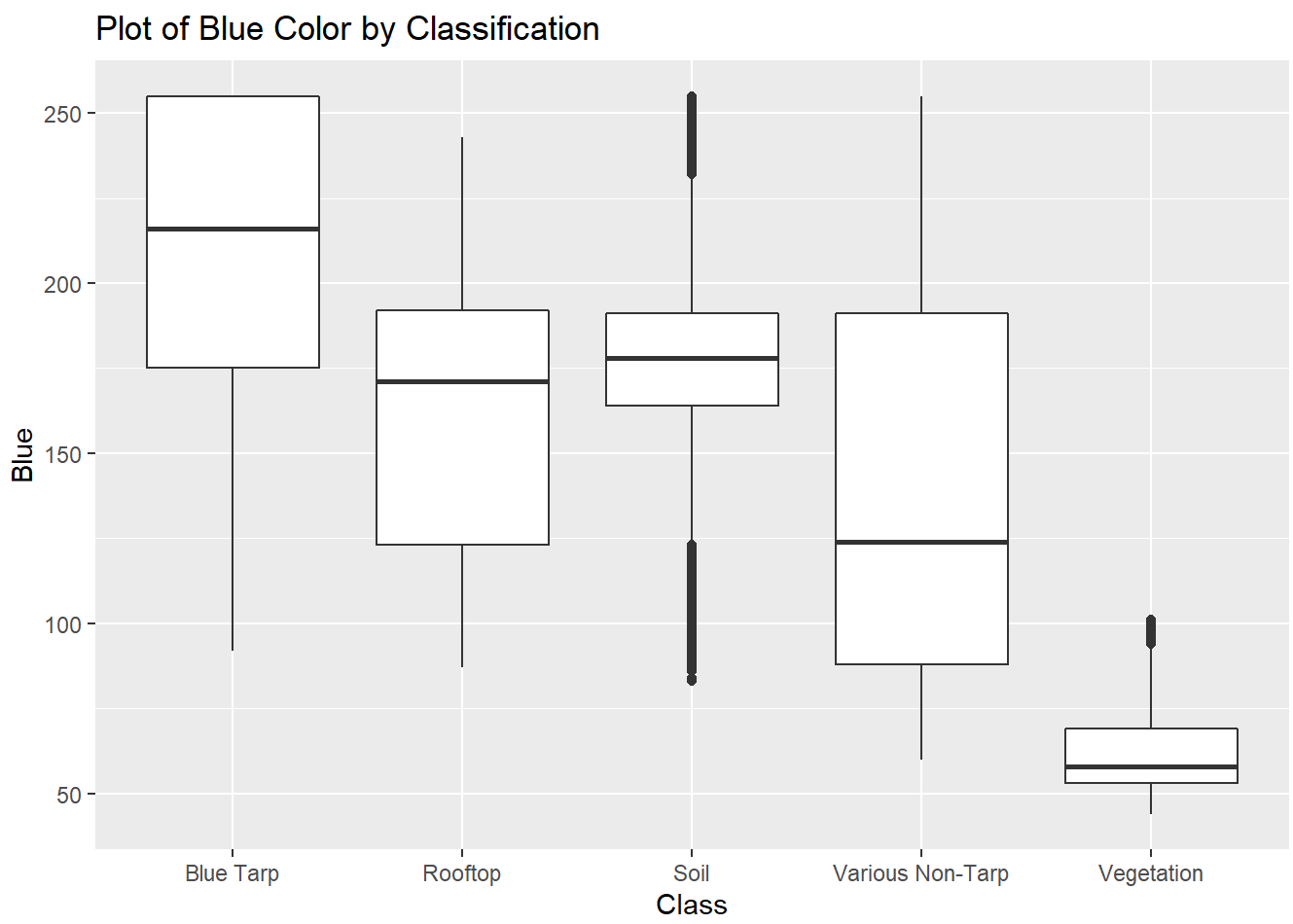
```
haiti %>%
  group_by(Class) %>%
  summarize(COR=cor(Blue,Green))
```

```
## # A tibble: 5 x 2
##   Class              COR
## * <chr>            <dbl>
## 1 Blue Tarp        0.926
## 2 Rooftop          0.985
## 3 Soil             0.840
## 4 Various Non-Tarp 0.959
## 5 Vegetation       0.905
```

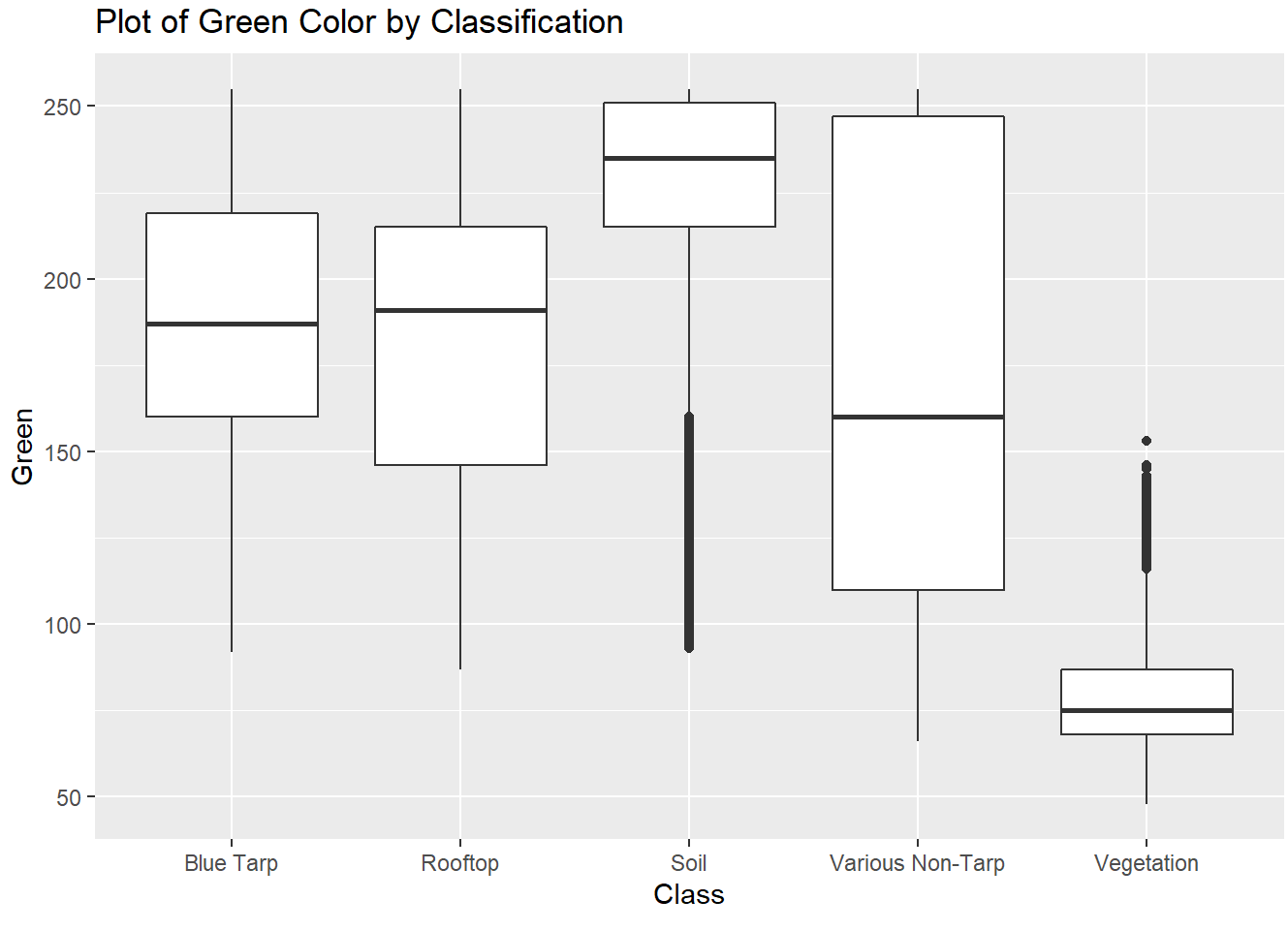
```
haiti %>%
  group_by(Class) %>%
  summarize(COR=cor(Red,Green))
```

```
## # A tibble: 5 x 2
##   Class              COR
## * <chr>            <dbl>
## 1 Blue Tarp        0.969
## 2 Rooftop          0.977
## 3 Soil             0.802
## 4 Various Non-Tarp 0.963
## 5 Vegetation       0.938
```

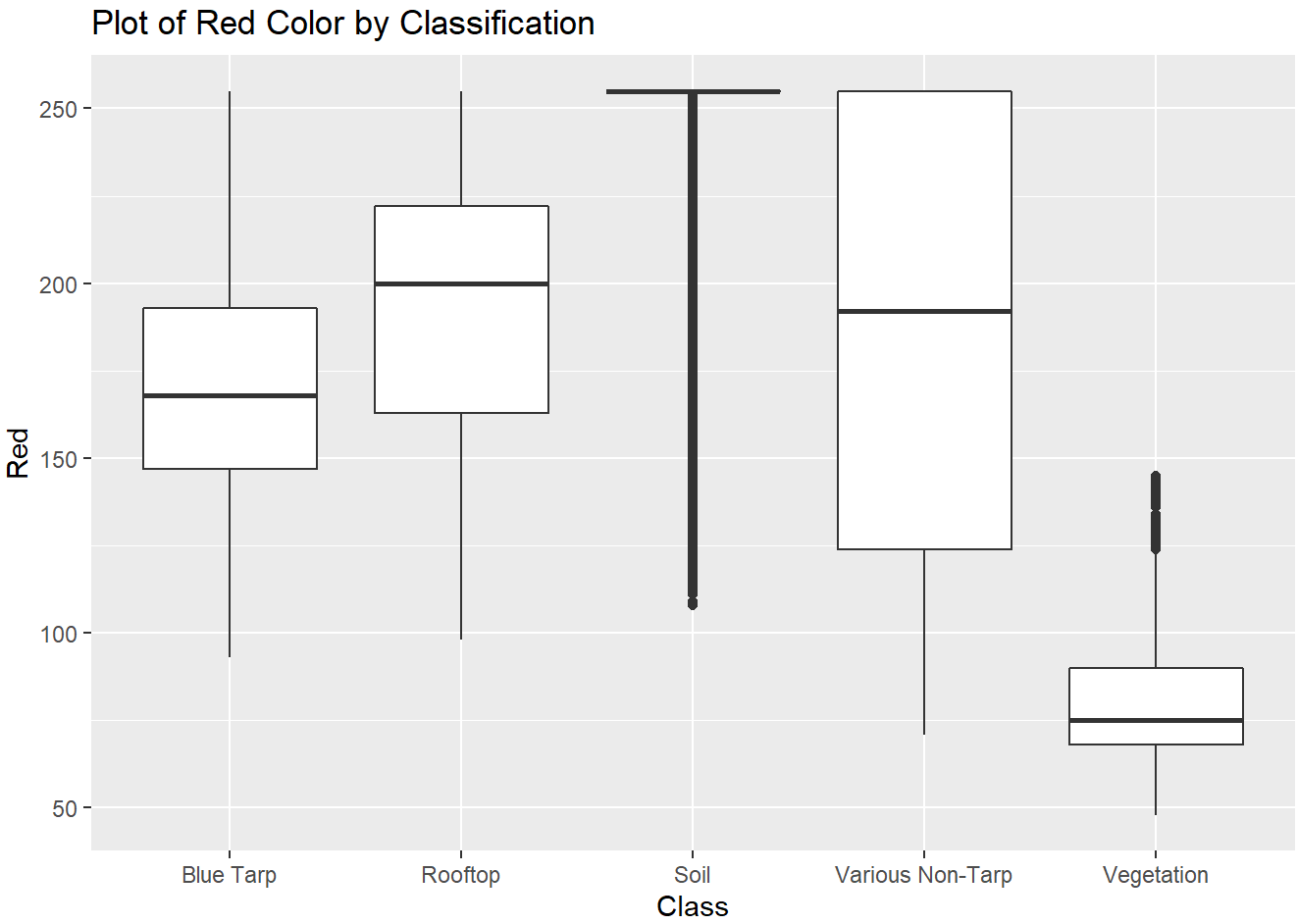
```
library(ggplot2)
ggplot(haiti, aes(x=Class, y=Blue)) +
  geom_boxplot() +  ggtitle("Plot of Blue Color by Classification")
```




```
ggplot(haiti, aes(x=Class, y=Green)) +
  geom_boxplot() + ggtitle("Plot of Green Color by Classification")
```



```
ggplot(haiti, aes(x=Class, y=Red)) +
  geom_boxplot() + ggtitle("Plot of Red Color by Classification")
```



Summary from EDA:

1. 2022 observations are BlueTarp or 3.2% of the observations.
2. The distribution of the “Blue” codes for “Blue Tarps” have the highest median of ~215 compared to the other classifications.
3. Vegetation has the lowest blue code values of code 100 or less.
4. RGB Variables are correlated
5. Red and Green have a higher correlation under BlueTarp classification (0.9690130), than Blue & Green (0.9257670), or Blue & Red (0.8594749)

3 Model Training

3.1 Test & Train Set-up

```
haiti$Class.Dummy <- 0
haiti$Class.Dummy <- ifelse(haiti$Class=="Blue Tarp",1,0)
attach(haiti)
```

```
## The following objects are masked from haiti (pos = 3):
##
##     Blue, Class, Green, Red
```

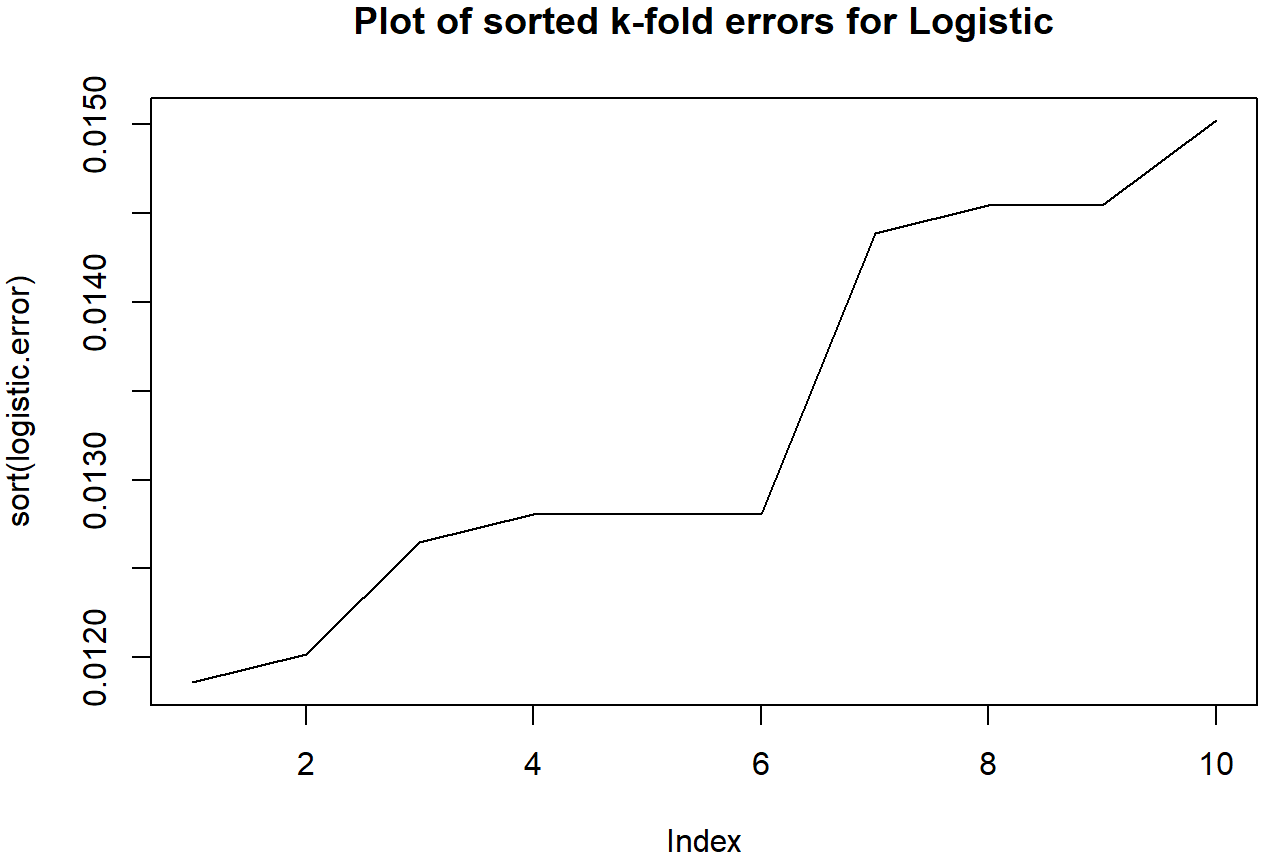
```
set.seed(914)
test20 = .20*nrow(haiti)
haiti.test = sample(nrow(haiti),test20 )  #vector of obs numbers
haiti.train = -haiti.test                 #removal of obs numbers for matrix
```

3.2 Logistic Regression

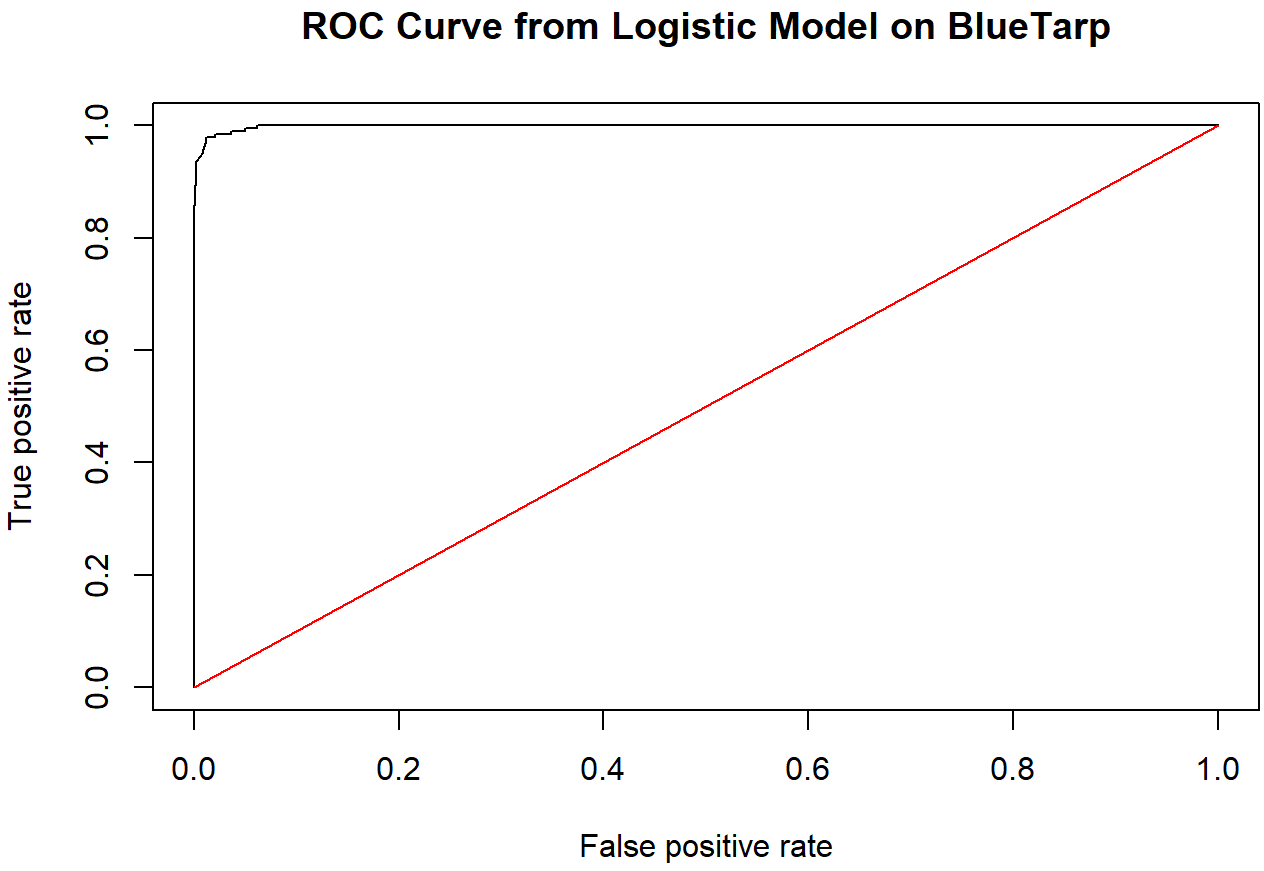
```
#k-fold 10

M = 10
n_test = as.integer(nrow(haiti)/M)

set.seed(552)

logistic.error = numeric(M)
for(m in 1:M) {
  # Create test and train split
  test = sample(nrow(haiti), n_test)
  train = -test
  # Fitting the model
  glm.predict = glm(Class.Dummy ~ Red + Green + Blue,
                family=binomial,
                data=haiti[train,])
  # Predictions
  p_hat = predict(glm.predict, haiti[test,], type = "response")
  cutoff.mn = mean(p_hat)
  cutoff.md = median(p_hat)
  cutoff = cutoff.mn + ((cutoff.mn - cutoff.md)/2)
  lm.cutoff = .034
  g_hat = ifelse(p_hat > cutoff, 1, 0)
  # Test Error
  logistic.error[m] = mean(haiti$Class.Dummy[test] != g_hat)
}
logistic.error
```

```
##  [1] 0.01265022 0.01280835 0.01454775 0.01502214 0.01185958 0.01280835
##  [7] 0.01201771 0.01438963 0.01280835 0.01454775
```

```
plot(sort(logistic.error), type = "l", main="Plot of sorted k-fold errors for Logistic")
```




```
#AUROC
library(ROCR)

#Apply to test data

p_hat = predict(glm.predict, haiti[test,], type = "response")

#Classification table for ROC
c_rate<-prediction(p_hat, haiti$Class.Dummy[test])
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from Logistic Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC
auc<-performance(c_rate, measure = "auc")
lm.auc <-auc@y.values[[1]]

##confusion matrix
lm.table <- table(haiti$Class.Dummy[test], p_hat>lm.cutoff)
lm.table
```

```
##
##     FALSE TRUE
##   0  6022  112
##   1     4  186
```

```
lm.applied <- cbind(haiti$Class.Dummy[test],p_hat>lm.cutoff)
```

3.2.1 Logistic Tuning

Since we are performing a standard logistic regression with no penalties or other hyperparameters, there aren't specific tuning parameters in this model.

3.2.2 Logistic Threshold Justification

I originally looked at the mean as the cut-off for the threshold so that it would vary between test samples. After reviewing the classifications tables I determined I would rather have 100% true positives at the cost of getting some false positives because it's better to think there is a tarp versus there is a tarp, but we didn't predict one to be

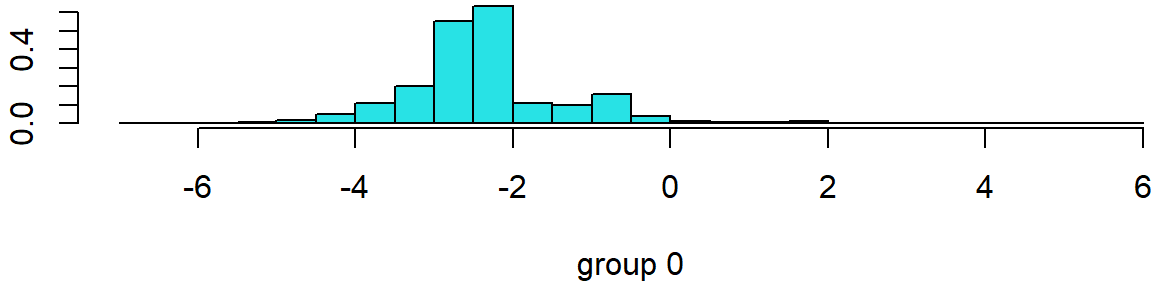
there. I iteratively increased the cutoff value from the mean and landed on .034 which reduces the Type I error, with the least amount of observations in the test set classified as not having a tarp, when in fact they do (n=4).

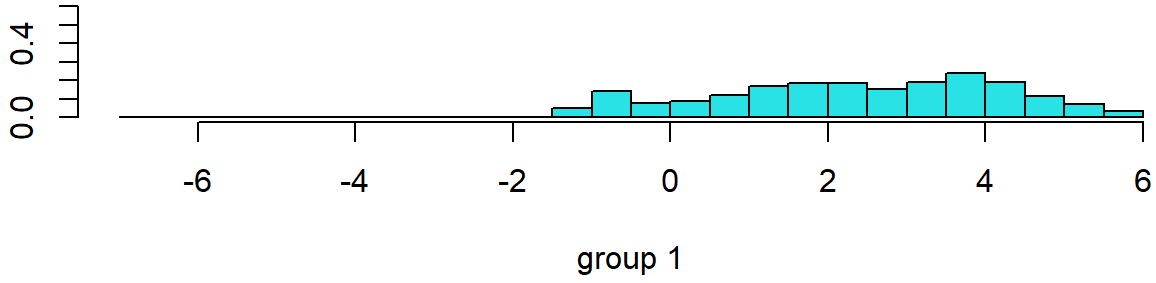
3.3 LDA

```
#k-fold 10

lda.error = numeric(M)
for(m in 1:M) {
  # Create test and train split
  haiti.test = sample(nrow(haiti), n_test)
  haiti.train = -test

  # Fitting the model
  lda.fit=lda(Class.Dummy~Red+Green+Blue, data=haiti,subset=haiti.train) #An index vector
          cases to be used in  training sample
  lda.fit
 # plot(lda.fit)

  # Predictions
  lda.pred = predict(lda.fit, haiti[haiti.test,])
  lda.class = lda.pred$class #predictions of BlueTarp on test data
  lda.cutoff = .015
  lda.class.adj <- ifelse(lda.pred$posterior[, 2] > lda.cutoff, "TRUE", "FALSE")  #tried b
         etween .01 & .035
  lda.class.adj <-lda.class.adj == "TRUE"
  # Test Error

  lda.table = table(haiti$Class.Dummy[haiti.test],lda.class.adj)


  lda.error[m] = mean(haiti$Class.Dummy[haiti.test] != lda.class.adj)
}
lda.error
```

```
##  [1] 0.01897533 0.02403542 0.02134725 0.02166350 0.02577483 0.01897533
##  [7] 0.02055661 0.02387729 0.02356104 0.01929159
```

```
plot(lda.fit)
```



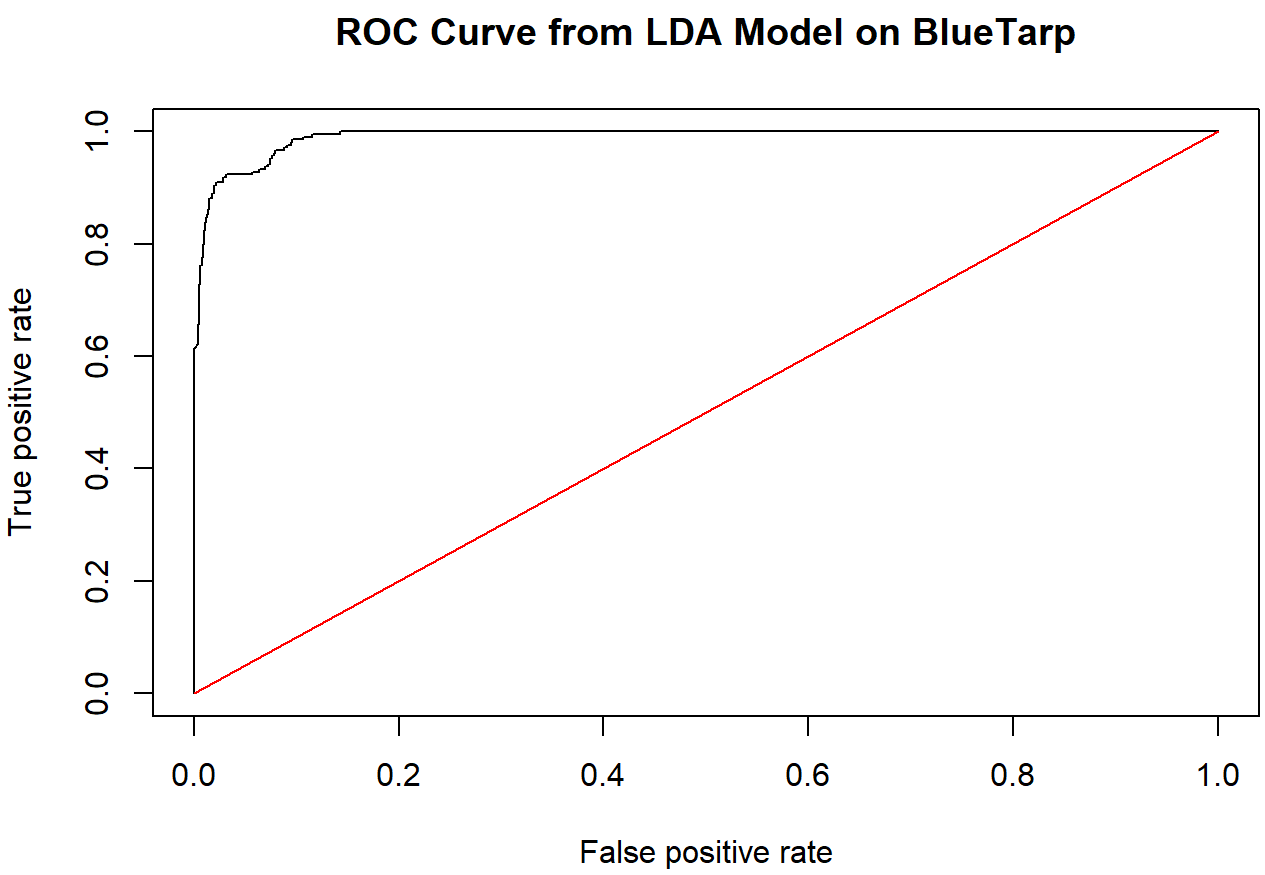



```
plot(sort(lda.error), type = "l", main="Plot of sorted k-fold errors for LDA")
```

```
lda.table
```

```
##    lda.class.adj
##     FALSE TRUE
##   0  6017   97
##   1    25  185
```

```
#AUROC
library(ROCR)

#Apply to test data

lda.pred = predict(lda.fit, haiti[haiti.test,])

#Convert posterirors to dataframe
lda.pred.post <-as.data.frame(lda.pred$posterior)

#Classification table for ROC
lda.btarp<-prediction(lda.pred.post[,2], as.data.frame(haiti$Class.Dummy[haiti.test]))
roc_data<-performance(lda.btarp, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from LDA Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```



```
#Find AUC (higher is better)
auc<-performance(lda.btarp, measure = "auc")
lda.auc = auc@y.values[[1]]

##confusion matrix
lda.table = table(haiti$Class.Dummy[haiti.test],lda.class.adj)
```

3.3.1 LDA Tuning

There aren't specific tuning parameters in this model.

3.3.2 LDA Threshold Justification

As with Logistic, I originally looked at the mean as the cut-off for the threshold so that it would vary between test samples. After reviewing the classifications tables and knowing I would rather have 100% true positives at the cost of getting some false positives because it's better to think there is a tarp versus there is a tarp, but we didn't predict one to be there. Through iterations, I decreased the cutoff value by .002 increments from the mean and landed on .015 which reduces the Type I error, with the least amount of observations in the test set classified as not having a tarp, when in fact they do.

3.4 QDA

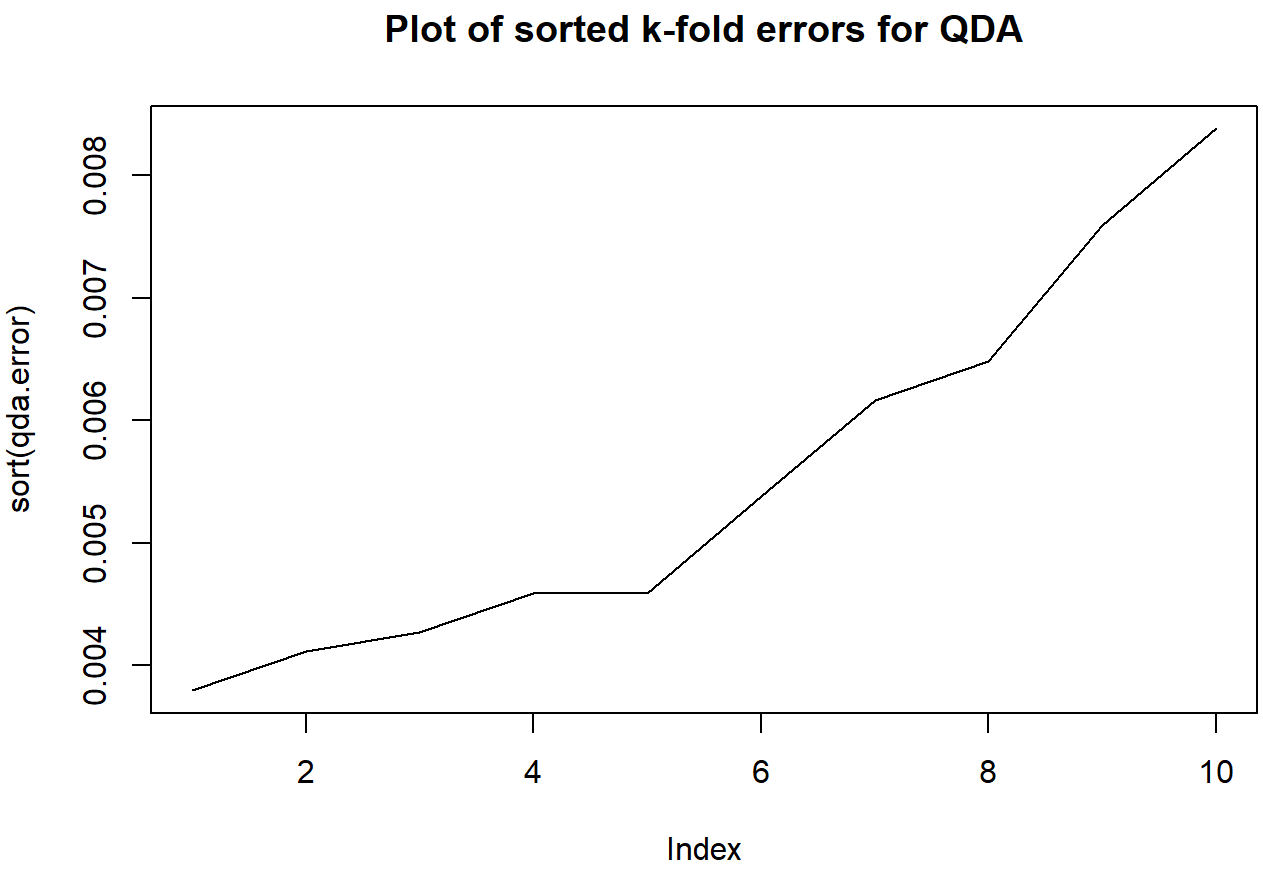
```
#k-fold 10



qda.error = numeric(M)
for(m in 1:M) {
  # Create test and train split
  haiti.test = sample(nrow(haiti), n_test)
  haiti.train = -test

  # Fitting the model
  qda.fit<- qda(Class.Dummy ~ Red + Green + Blue, data=haiti,subset=haiti.train)
  qda.fit



  # Predictions
  qda.class = predict(qda.fit,haiti[haiti.test,] )$class #predictions of BlueTarp on test
          data
  qda.post = predict(qda.fit,haiti[haiti.test,] )$posterior



  #Threshold Adjustments
  qda.cutoff = .5
  qda.class.adj <- ifelse(qda.post[, 2] > qda.cutoff, "TRUE", "FALSE")
  qda.class.adj <-qda.class.adj == "TRUE"

  # Test Error

  qda.table = table(haiti$Class.Dummy[haiti.test],qda.class)
  qda.table.adj = table(haiti$Class.Dummy[haiti.test],qda.class.adj)

  qda.error[m] = mean(haiti$Class.Dummy[haiti.test] != qda.class)
 # qda.error.adj[m] = mean(haiti$Class.Dummy[haiti.test] != qda.class.adj)

  }
qda.error
```

```
##  [1] 0.007590133 0.004585705 0.004111322 0.005376344 0.006483238 0.004585705
##  [7] 0.006166983 0.008380772 0.004269450 0.003795066
```

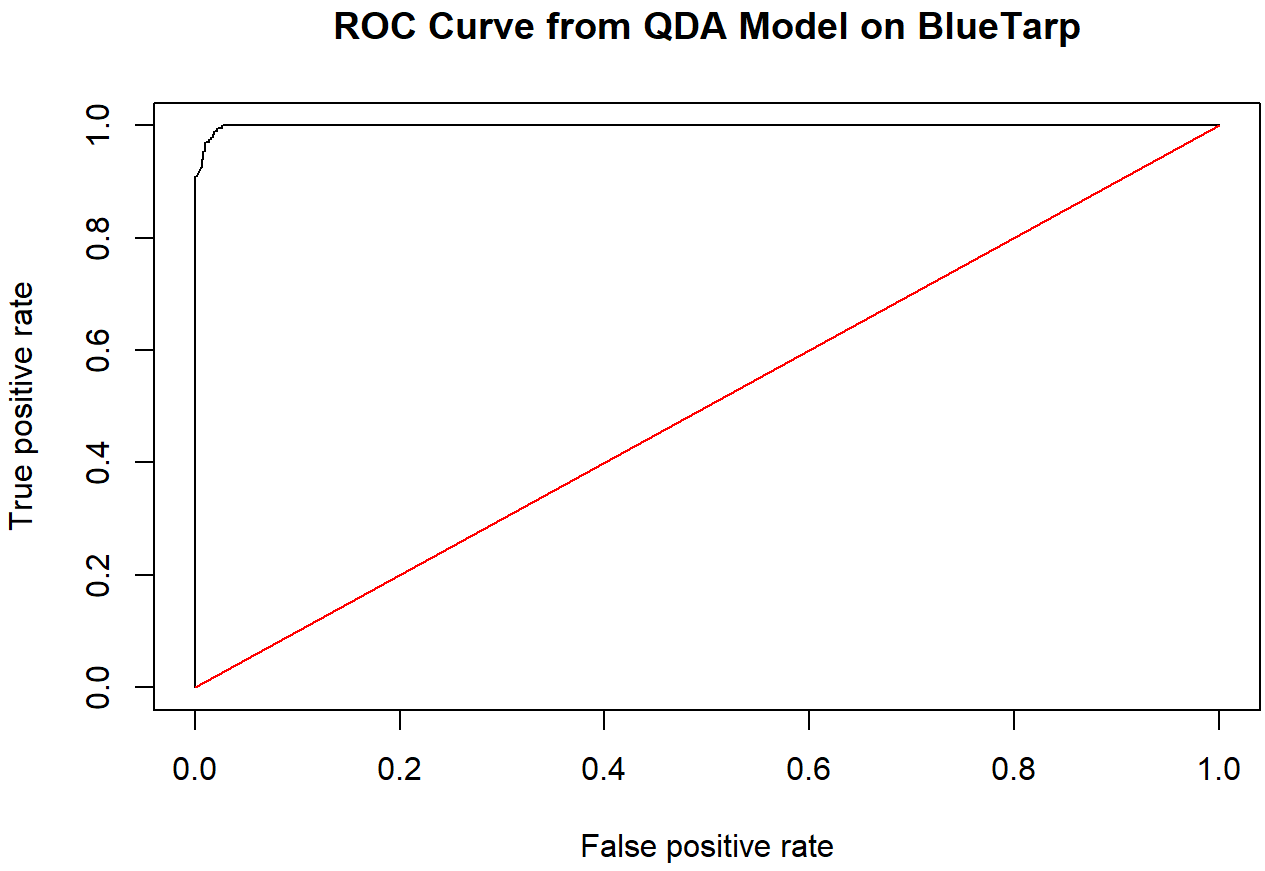
```
qda.table
```

```
##    qda.class
##        0    1
##   0 6122    1
##   1   23  178
```

```
#qda.table.adj


plot(sort(qda.error), type = "l", main="Plot of sorted k-fold errors for QDA")
```




```
#AUROC
library(ROCR)

#Apply to test data

qda.class = predict(qda.fit,haiti[haiti.test,] )$class #predictions of BlueTarp on test da
        ta
qda.pred = predict(qda.fit,haiti[haiti.test,] )

#Classification table for ROC
qda.btarp<-prediction(qda.pred$posterior[,2], as.data.frame(haiti$Class.Dummy[haiti.tes
        t]))

roc_data<-performance(qda.btarp, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from QDA Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC (higher is better)
auc<-performance(qda.btarp, measure = "auc")
qda.auc = auc@y.values[[1]]

##confusion matrix
qda.table = table(haiti$Class.Dummy[haiti.test],qda.class)


#from: https://rpubs.com/Kushan/295412
```

3.4.1 QDA Tuning

There aren't specific tuning parameters in this model.

3.4.2 QDA Threshold Justification

After reviewing the classifications tables and knowing I would rather have 100% true positives at the cost of getting some false positives I chose the default cutoff of .5 for the QDA model which reduced the Type I error, with the least amount of observations in the test set classified as not having a tarp, when in fact they do.

3.5 KNN

```
library(class)
library(ROCR)
library(ISLR)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

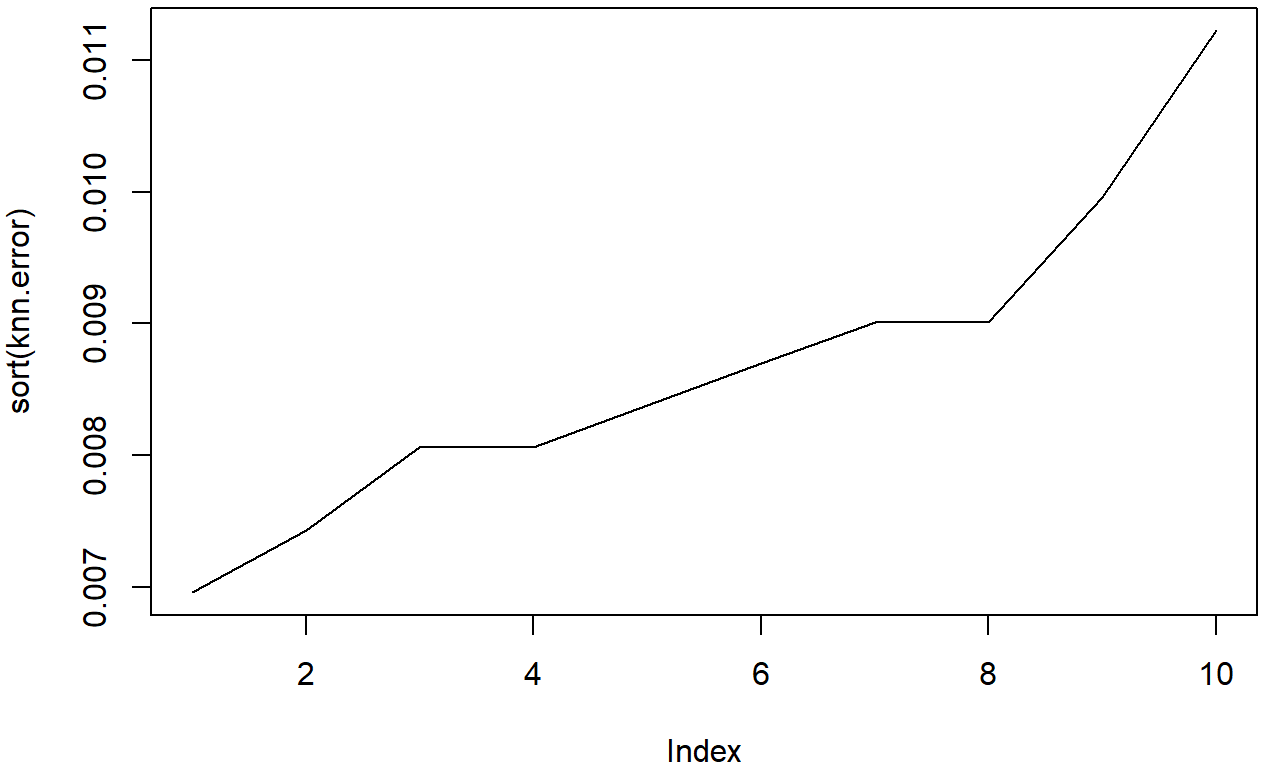
```
#k-fold 10
M = 10
n_test = as.integer(nrow(haiti)/M)

knn.error = numeric(M)
for(m in 1:M) {
  # Create test and train split
  haiti.test = sample(nrow(haiti), n_test)
  haiti.train = -haiti.test

# Fitting the model & Predictions
knn_mod = knn3(Class.Dummy~Red+Green+Blue, data=haiti[haiti.train,], k = 5) #knn model
knn.pred <- predict(knn_mod, haiti[haiti.test,], type = "prob") #predict onto holdout


knn.prob=knn.pred[,2]
#head(knn.prob)

knn.pred.class <- knn.prob > 0.05
knn.pred.class <- ifelse(knn.prob > 0.05,1,0)

# Test Error

test.Class.Dummy <- haiti[haiti.test,"Class.Dummy"]
knn.table = table(test.Class.Dummy,knn.pred.class)
knn.table

knn.error[m] = mean(test.Class.Dummy != knn.pred.class)


}

print("Test Error")
```

```
## [1] "Test Error"
```

```
knn.error
```

```
##  [1] 0.008380772 0.008064516 0.008697027 0.008064516 0.006957622 0.011227071
##  [7] 0.009962049 0.007432005 0.009013283 0.009013283
```

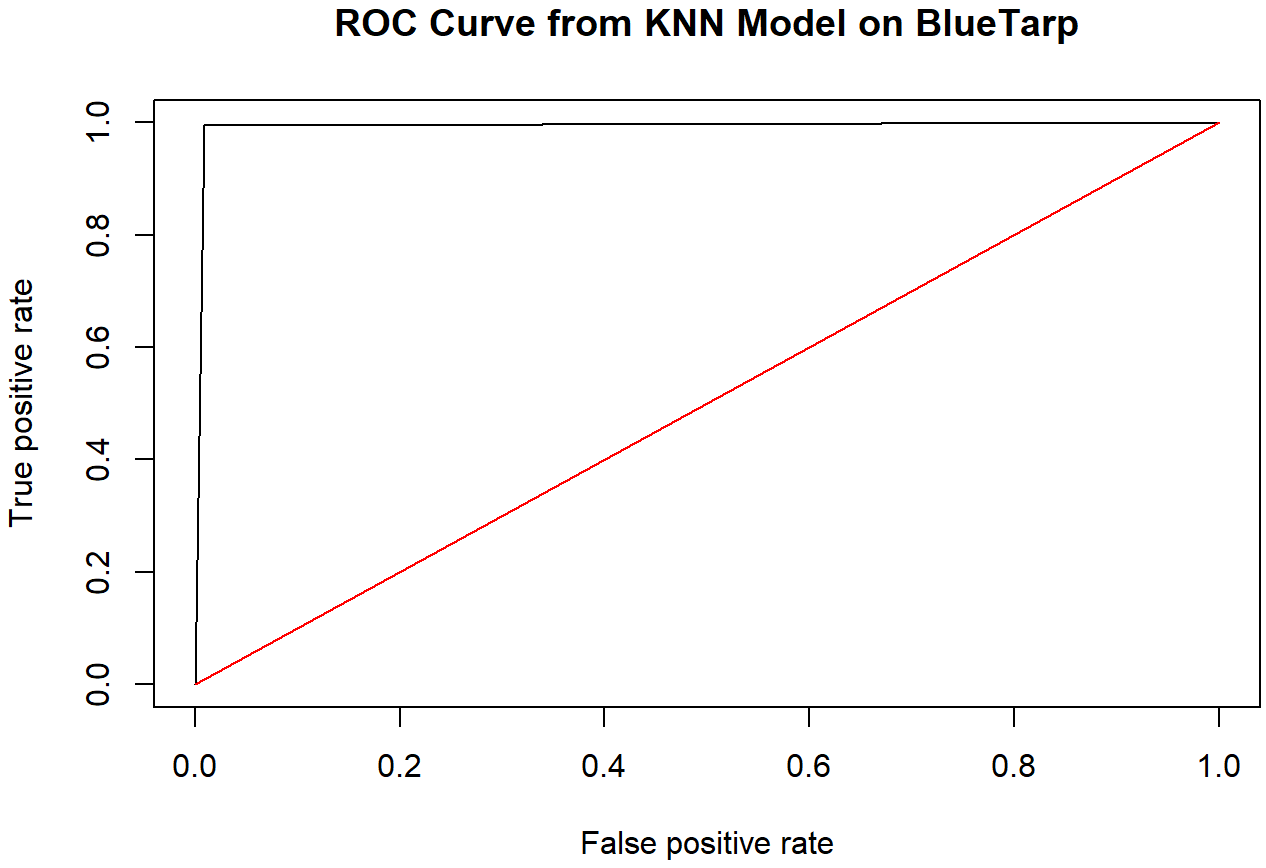
```
plot(sort(knn.error), type = "l", main="Plot of sorted k-fold errors for KNN")
```




```
knn.table
```

```
##                 knn.pred.class
## test.Class.Dummy    0    1
##                0 6077   56
##                1    1  190
```

```
#AUROC
#https://rstudio-pubs-static.s3.amazonaws.com/16444_caf85a306d564eb490eebdbaf0072df2.html

#Classification table for ROC
c_rate<-prediction(knn.pred.class, test.Class.Dummy )
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from KNN Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC
auc<-performance(c_rate, measure = "auc")
knn.auc <-auc@y.values[[1]]
```

3.5.1 Tuning Parameter k

How were tuning parameter(s) selected? What value is used? Plots/Tables/etc.

I used a classification table to determine what tuning parameter to use for each k.. I compared k=1, k=3, k=5 and k=10 for a single training and test dataset (prior to k-fold 10). knn k=5 has best test error rate.

3.6 Penalized Logistic Regression (ElasticNet)

```
library(glmnet)



#k folds (10)

set.seed(21)
plr.error = numeric(M)
for(m in 1:M) {
  # Create test and train split
  haiti.test = sample(nrow(haiti), n_test)
  haiti.train = -test
  train.x <- cbind(Red, Green, Blue )[haiti.train, ] #matrix containing predictors associa
         ted with training data
  test.x <- cbind(Red, Green, Blue)[haiti.test, ] #matrix containing predictors associated
         with testing data
  train.Class.Dummy = Class.Dummy[haiti.train] #vector containing class labels for the tra
         ining observations
  BlueTarp = Class.Dummy[haiti.test]



  # Fitting the model & Predictions
  fit <- glmnet(train.x,train.Class.Dummy)
 cvfit <- cv.glmnet(haiti.matrix[ ,2:4],as.vector(Class.Dummy),nfolds=10)
  predict.BlueTarp <- predict(fit, newx=train.x,  s = cvfit$lambda.min)

  # Test Error

  summary(predict.BlueTarp, mean()) #to get cutoff
  plr.cutoff = mean(predict.BlueTarp)

  classify.BlueTarp <- ifelse(predict.BlueTarp > plr.cutoff, 1, 0)

  plr.table = table(train.Class.Dummy,classify.BlueTarp)

  plr.error[m] = mean(train.Class.Dummy != classify.BlueTarp)

}

print("Summary of Predicted Values of Blue Tarp")
```

```
## [1] "Summary of Predicted Values of Blue Tarp"
```

```
  summary(predict.BlueTarp, mean()) #to get cutoff
```

```
##        1
##  Min.   :-0.37368
##  1st Qu.:-0.01961
##  Median : 0.01283
##  Mean   : 0.03219
##  3rd Qu.: 0.04743
##  Max.   : 0.73423
```

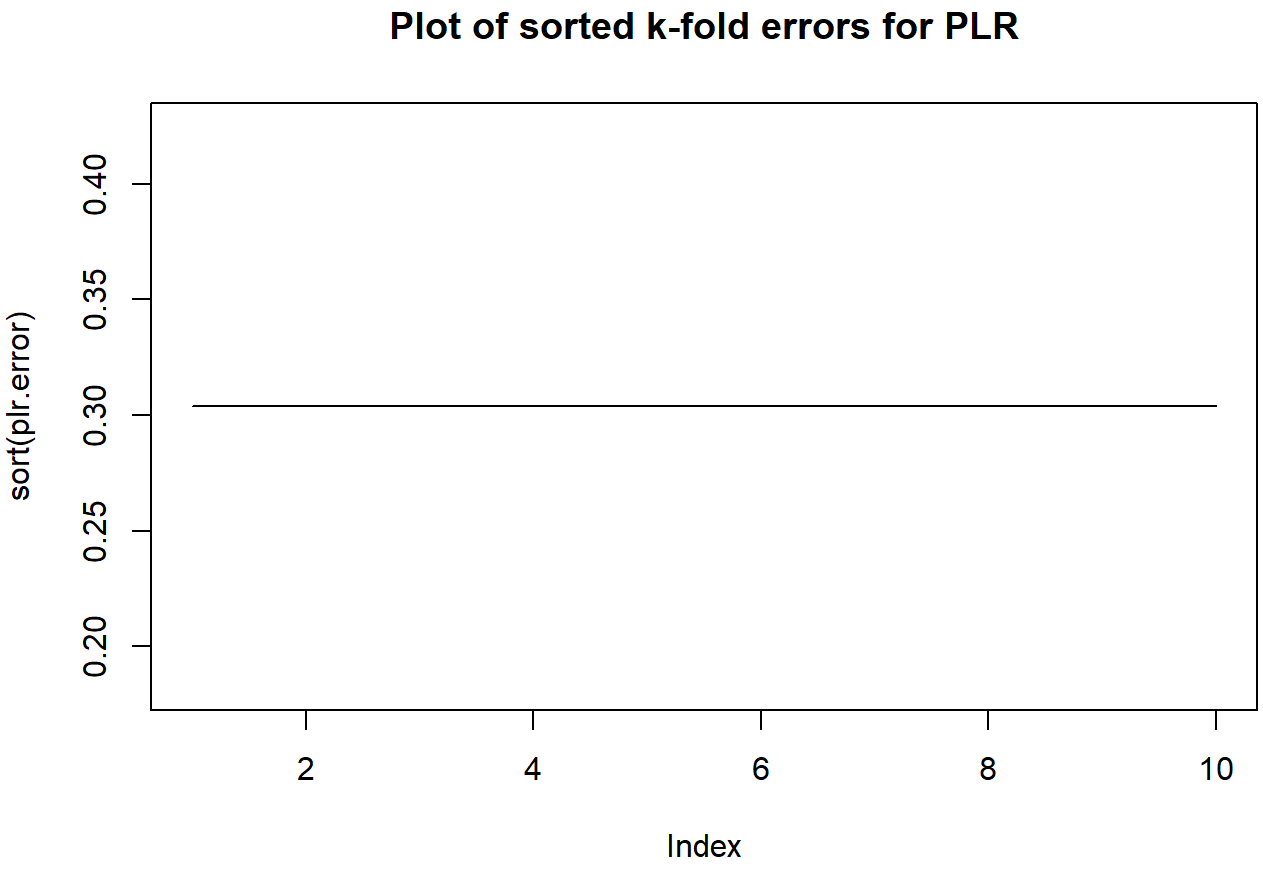
```
plr.error
```

```
##  [1] 0.3037757 0.3037757 0.3037757 0.3037757 0.3037757 0.3037757 0.3037757
##  [8] 0.3037757 0.3037757 0.3037757
```

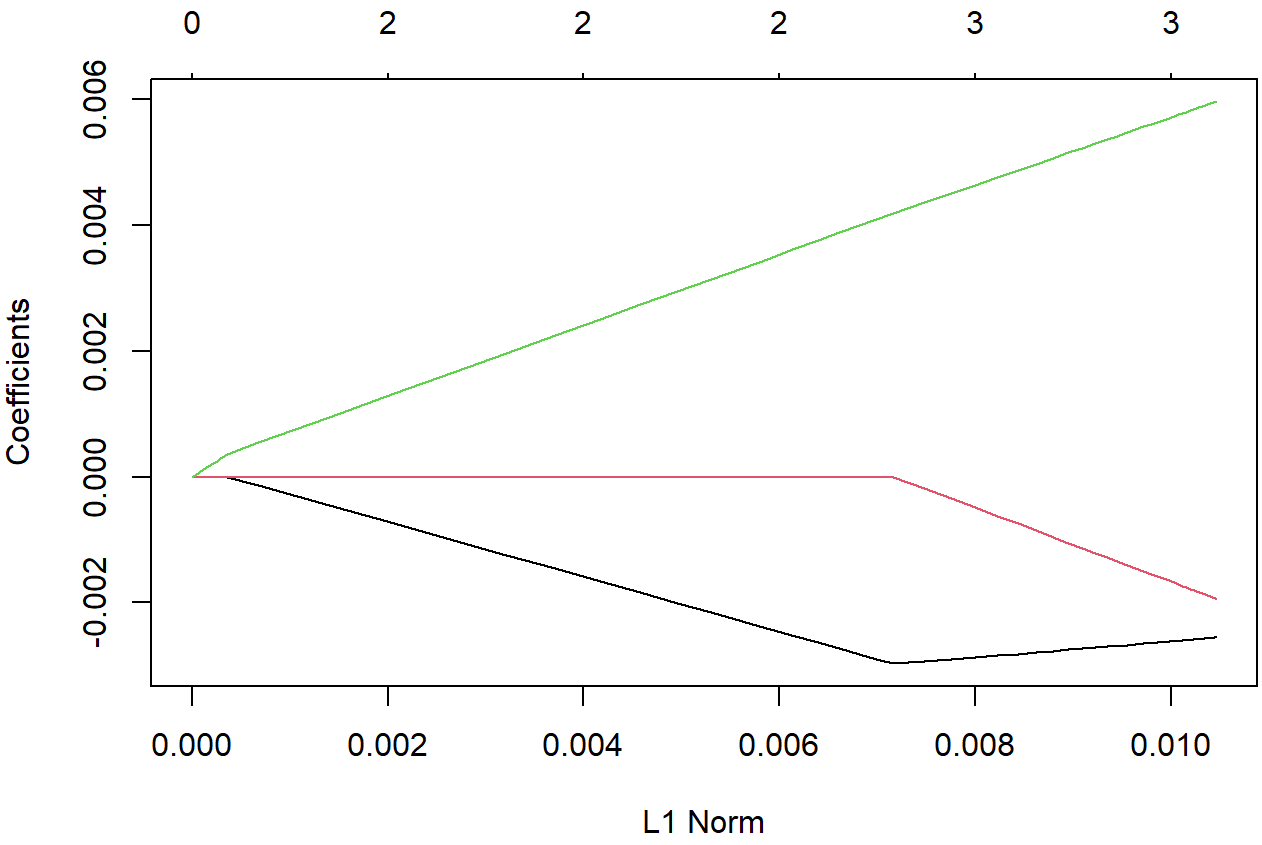
```
plr.table
```

```
##                  classify.BlueTarp
## train.Class.Dummy     0     1
##                 0 37795 17290
##                 1     0  1832
```

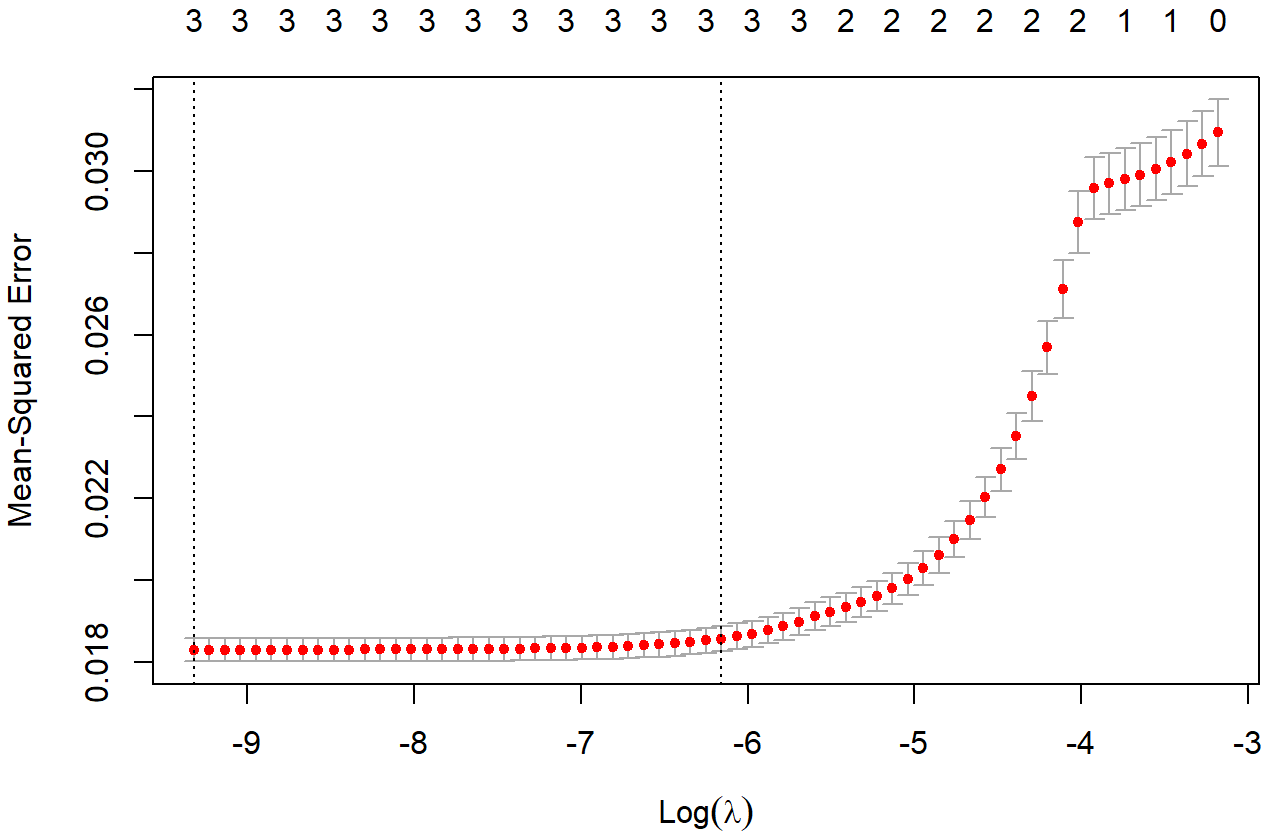
```
plot(sort(plr.error), type = "l", main="Plot of sorted k-fold errors for PLR")
```



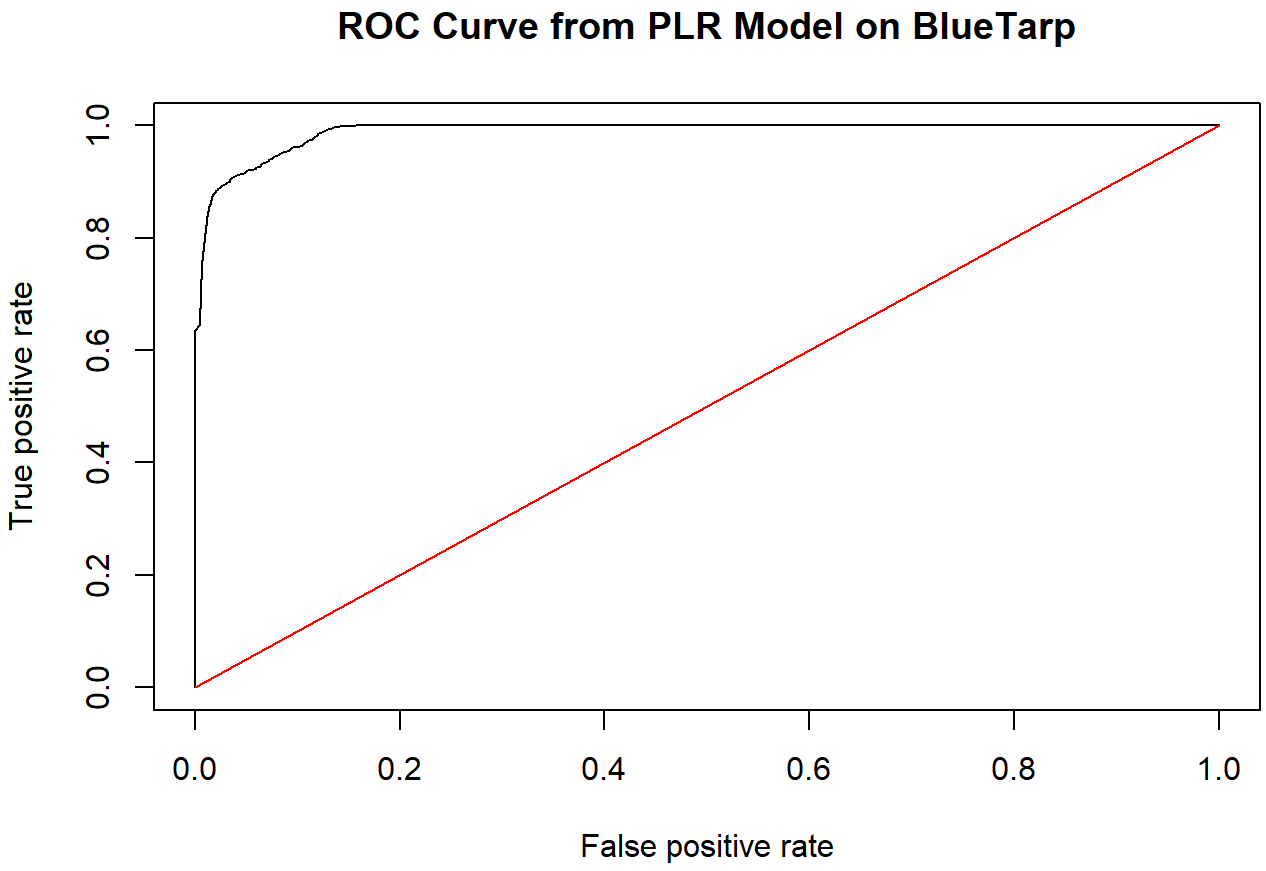
```
plot(fit)
```




```
plot(cvfit)
```




```
#Classification table for ROC
c_rate<-prediction(predict.BlueTarp, train.Class.Dummy)
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from PLR Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC
plr.auc<-performance(c_rate, measure = "auc")
plr.auc <-auc@y.values[[1]]
```

3.6.1 Tuning Parameters

I looked at the fit (fit <- glmnet(haiti.matrix[,2:4],as.vector(Class.Dummy)). It shows from left to right the number of nonzero coefficients (Df), the percent (of null) deviance explained (%dev) and the value of λ (Lambda). So I chose a lambda that includes 3 df. The best alpha and lambda values are those values that minimize the cross-validation error.

3.6.2 Threshold Selection

Thresholds were selected by taking the mean predicted value of the training data. Since the true percentage of Blue Tarps is low, I would not want to use a cutoff of 50%. I also wanted to consider that I would rather predict there is a tarp when there wasn't (100% True, some False), than wrongly predict that there was not a tarp when there actually was. So my selections looked for that to be the case.

3.6.3 posterior probability cutoff tuning

In general, instead of arbitrarily choosing λ = 4, it would be better to use cross-validation to choose the tuning parameter λ. We can do this using # the built-in cross-validation function, cv.glmnet(). By default, the function cv.glmnet() performs ten-fold cross-validation, though this can be changed using the argument nfolds.

3.7 Random Forest

```
#Bagging and Random Forests

library(randomForest)
```

```
## randomForest 4.6-14
```

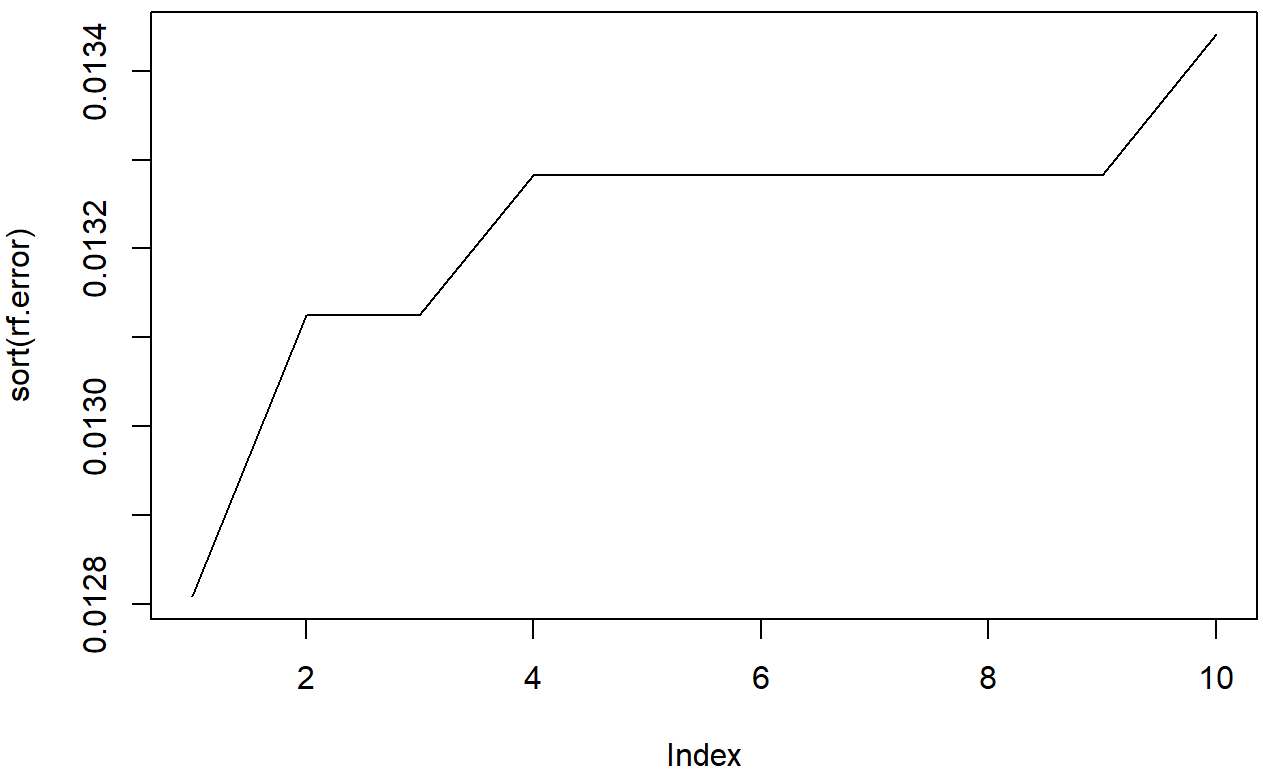
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)
rf.error = numeric(M)
M = 10
n_test = as.integer(nrow(haiti)/M)

for(m in 1:M) {
  # Create test and train split
  haiti.test = sample(nrow(haiti), n_test)
  haiti.train = -test

bag.haiti <- randomForest(haiti$Class.Dummy~Red+Red^2+Green+Green^2+Blue+Blue^2+Blue^3,data=haiti, subset=haiti.train,mtry=3,importance=TRUE, ntrees=500) #2 methods to decorrelate.
#1) bagging :pull random samples for the tree you are going to create
#2) split in each tree take subset of # of variables that are available
#mtry is the number of features we are going to try (variables)
#importance TRUE means keep importance values
bag.haiti
yhat.bag <-predict(bag.haiti, newdata=haiti[-haiti.train,])

yhat.class <- 0
yhat.class <- ifelse(yhat.bag > mean(yhat.bag),1,0)
rf.cutoff = mean(yhat.bag)

haiti.test=haiti[-haiti.train,"Class.Dummy"]

rf.table = table(haiti.test,yhat.class)
rf.error[m] = mean(haiti.test != yhat.class)
}


plot(sort(rf.error), type = "l", main="Plot of sorted k-fold errors for Random Forest")
```




```
rf.table
```

```
##           yhat.class
## haiti.test    0    1
##          0 6050   84
##          1    0  190
```

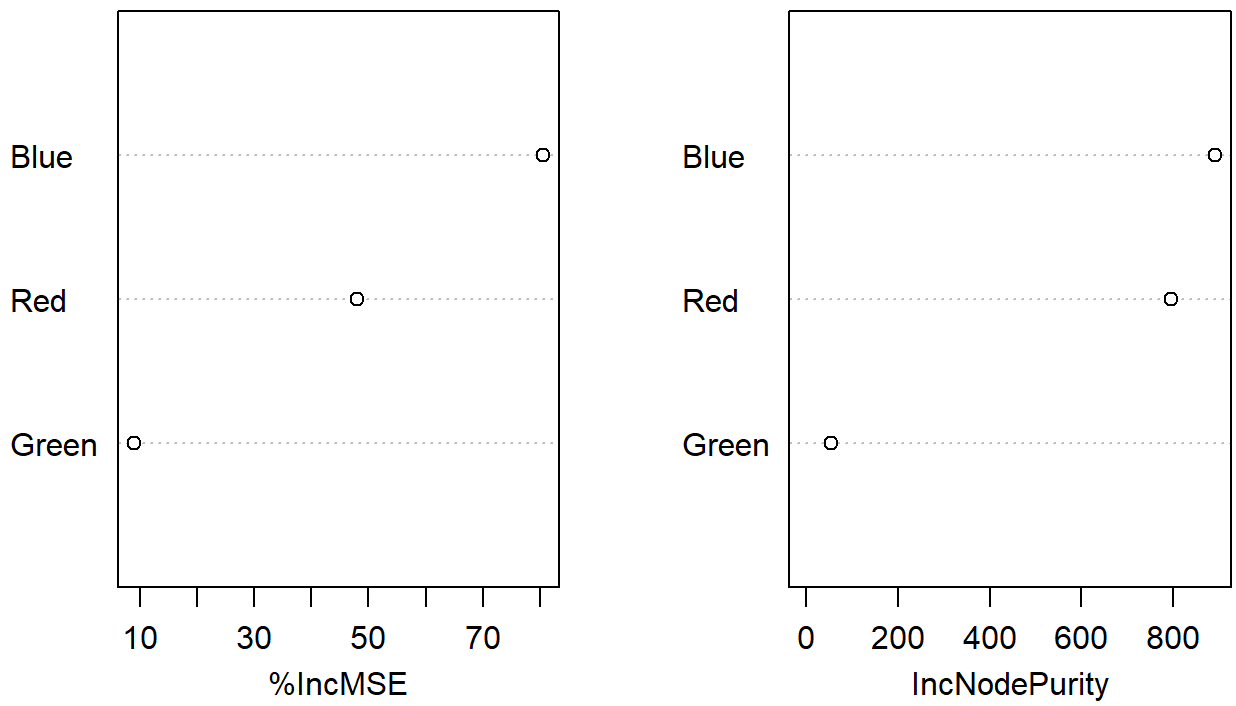
```
rf.error
```

```
##  [1] 0.01280835 0.01312460 0.01312460 0.01328273 0.01328273 0.01344086
##  [7] 0.01328273 0.01328273 0.01328273 0.01328273
```

```
importance(bag.haiti)
```

```
##         %IncMSE IncNodePurity
## Red   47.942369     795.32616
## Green  8.948977      54.45638
## Blue  80.508837     890.34802
```

```
varImpPlot(bag.haiti)
```




```
haiti.Class.Dummy <- as.factor(haiti$Class.Dummy)


################  Use best model for roc curve

library(randomForest)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#https://stackoverflow.com/questions/46124424/how-can-i-draw-a-roc-curve-for-a-randomforest-model-with-three-classes-in-r

haiti.Class.Dummy <- as.factor(haiti$Class.Dummy)

bag.haiti <- randomForest(haiti.Class.Dummy~Red+Red^2+Green+Green^2+Blue+Blue^2+Blue^3,data=haiti, mtry=3,importance=TRUE, ntrees=500)

bag.haiti
```

```
##
## Call:
##  randomForest(formula = haiti.Class.Dummy ~ Red + Red^2 + Green +      Green^2 + Blue + Blue^2 + Blue^3, data = haiti, mtry = 3,      importance = TRUE, ntrees = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.31%
## Confusion matrix:
##       0    1 class.error
## 0 61139   80 0.001306784
## 1   115 1907 0.056874382
```

```
yhat.bag <-predict(bag.haiti, newdata=haiti)
yhat.class <-as.data.frame(yhat.bag)

rf.table.ho <-table(haiti$Class.Dummy,yhat.bag) #(actual, predicted)
rf.table.ho
```

```
##    yhat.bag
##         0     1
##   0 61217     2
##   1    24  1998
```

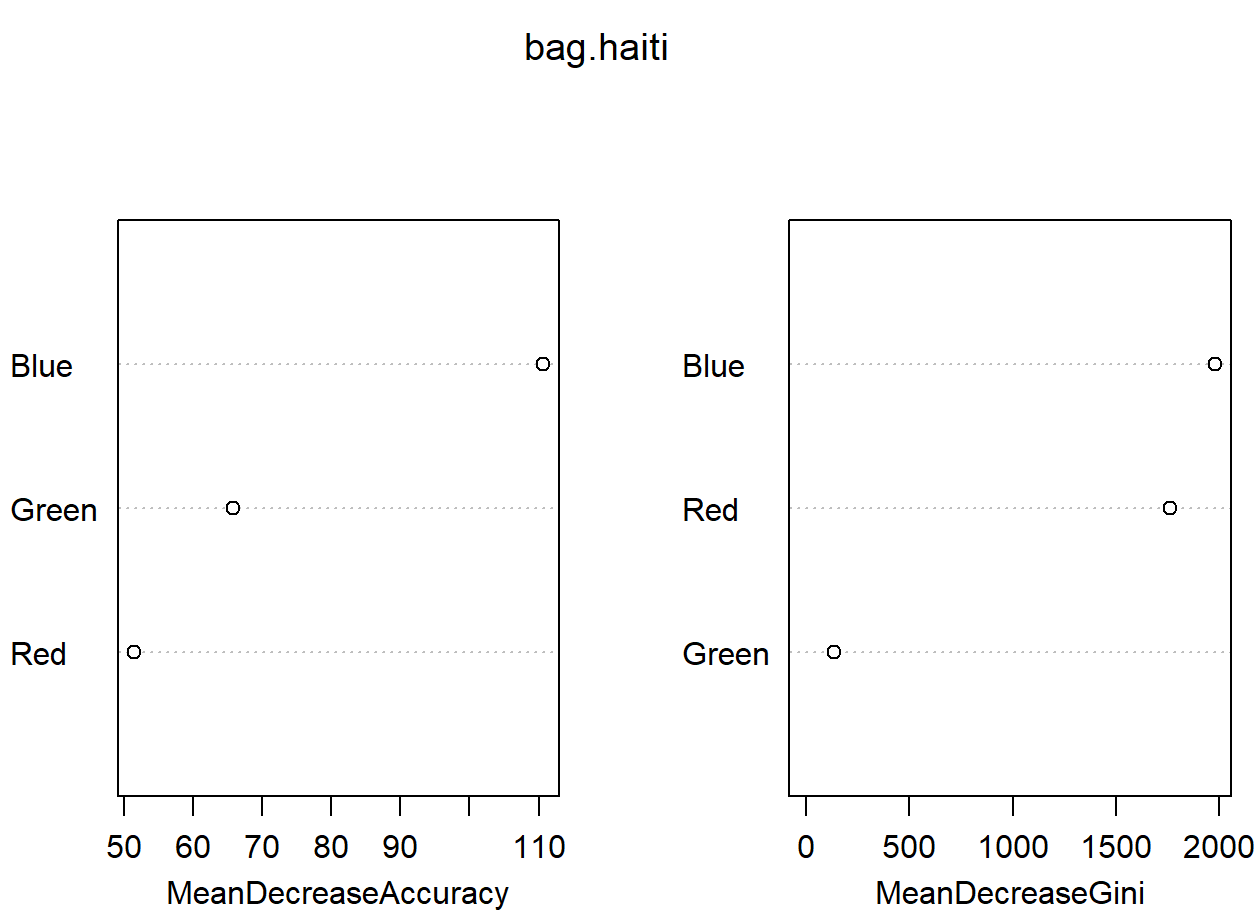
```
rf.error = mean(haiti$Class.Dummy != yhat.class$yhat.bag)
rf.error
```

```
## [1] 0.0004111257
```

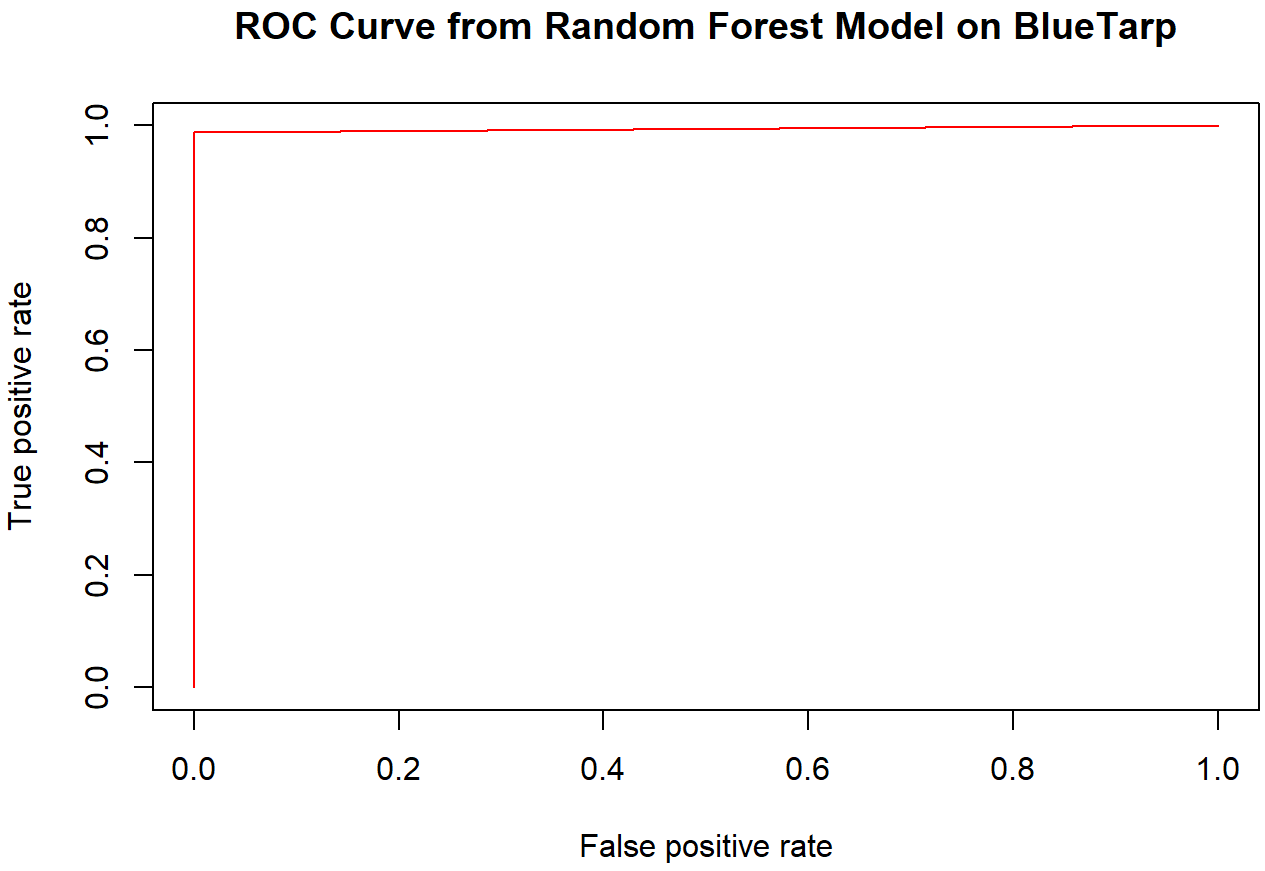
```
importance(bag.haiti)
```

```
##                0        1 MeanDecreaseAccuracy MeanDecreaseGini
## Red    46.869606 148.5860             51.49120         1759.953
## Green   7.990626  66.9282             65.84147          137.716
## Blue  101.206782 713.6488            110.59394         1977.025
```

```
varImpPlot(bag.haiti)
```




```
#Classification table for ROC
predictions <- as.numeric(predict(bag.haiti, haiti, type="response"))
pred <- prediction(predictions, haiti$Class.Dummy)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, col=rainbow(10), main="ROC Curve from Random Forest Model on BlueTarp")
```




```
rf.auc <- performance(pred,measure="auc")
rf.auc <-auc@y.values[[1]]
```

3.7.1 Tuning Parameters for Random Forest

I looked at different ntrees = 150, 500, and 1000. There was no difference in 500 & 1000 ntrees. At n=tree 150, the classification table added 1 observation as a False Negative, and I want to stay away from that. So I settled on ntree = 500 Without many features to choose from, mtry = 3 made the most sense.

3.8 Support Vector Machines (allows us to use different kernels other than linear)

```
# Gamma explains how far the influence of a single training example reaches. When gamma is
        very small, the model is too constrained and cannot capture the complexity or "sh
        ape" of the data.
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

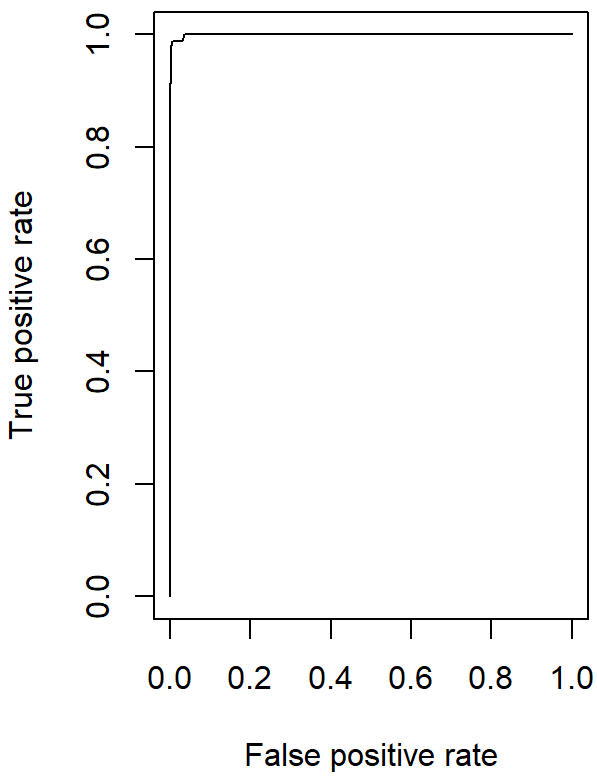
```
set.seed(914)
test20 = .20*nrow(haiti)
haiti.test = sample(nrow(haiti),test20 )  #vector of obs numbers
haiti.train = -haiti.test                 #removal of obs numbers for matrix



svmfit=svm(Class.Dummy~Red+Green+Blue+Blue^2, data=haiti[haiti.train,],kernel="radial",gam
        ma=2,cost=1, scale=TRUE)
plot(svmfit, haiti[haiti.train,])
summary(svmfit)
```

```
##
## Call:
## svm(formula = Class.Dummy ~ Red + Green + Blue + Blue^2, data = haiti[haiti.train,
##     ], kernel = "radial", gamma = 2, cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  3793
```

```
fitted=attributes(predict(svmfit,haiti,decision.values = TRUE)) $decision.values
pred <- predict(svmfit,haiti)

#table(true=holdout.all[,"Class.Dummy"],pred=predict(tune.out$best.model,newdata=haiti[hai
        ti.test,]))
# svm.table.ho = table(true=holdout.all$Class.Dummy,pred=fitted)

svm.table <- table(haiti$Class.Dummy, pred)#(actual, predicted)
#svm.table



dat.te=data.frame(x=haiti[,2:4], y=as.factor(haiti$Class.Dummy ))
pred.te=predict (svmfit , newdata =dat.te)

#need to classify raw probabilities
svm.cutoff = .001
svm.class <- ifelse(pred.te > svm.cutoff, 1, 0)
svm.table <- table(dat.te$y,svm.class) #(actual, predicted)
svm.table
```

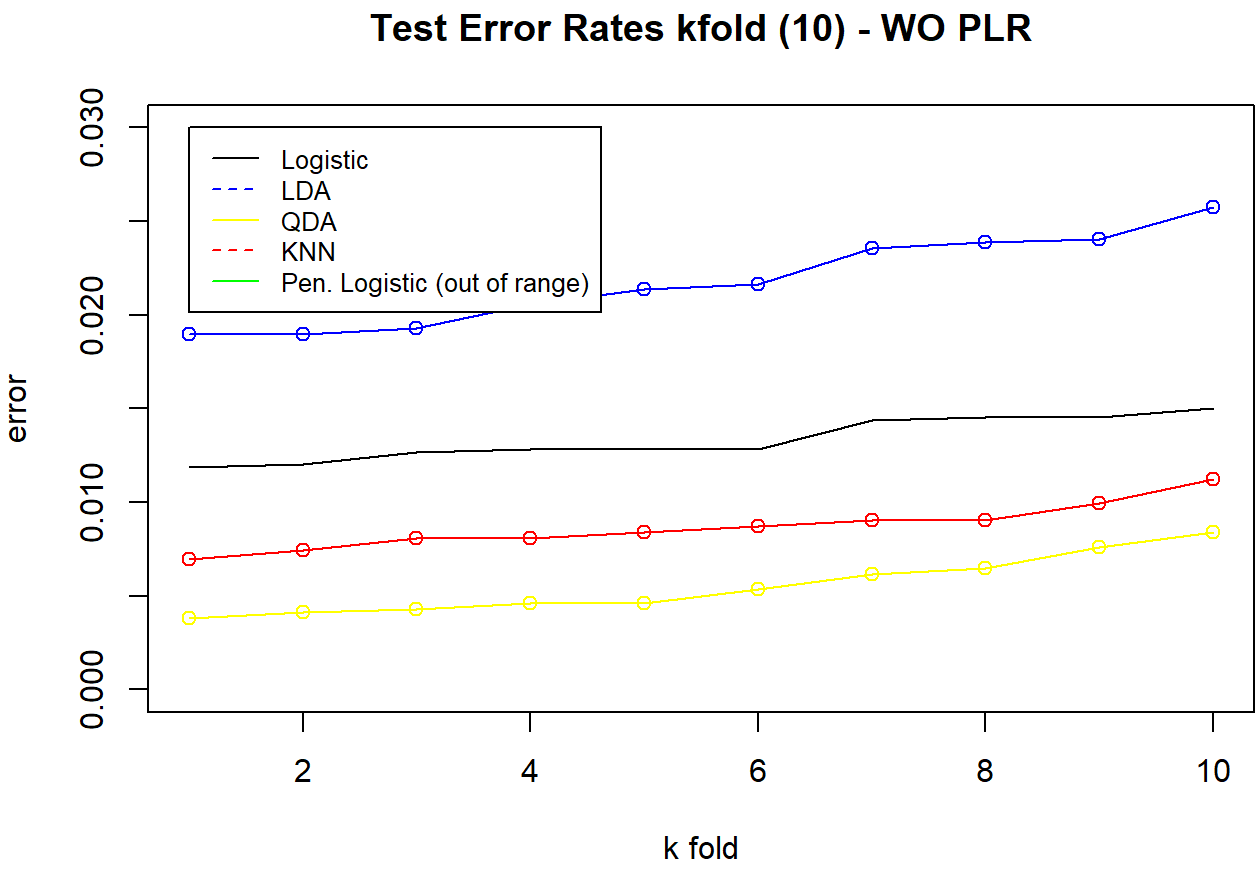
```
##    svm.class
##         0     1
##   0 43739 17480
##   1     0  2022
```

```
#ROC Curves
library (ROCR)
rocplot=function(pred.te,truth, ...) {
  predob = prediction(pred.te,truth)
  perf = performance(predob, "tpr" ,"fpr") #ignore the axes labels!! had to switch fpr and
         tpr to make plot correct
  plot(perf,...)}


par(mfrow=c(1,2)) #create one row two columns of figures to plot

rocplot(fitted, haiti[,"Class.Dummy"],main="ROC Curve from SVM Model on BlueTarp")
predob = prediction(pred.te,dat.te$y)
perf_val <- performance(predob,"auc")
svm.auc <-auc@y.values[[1]]
```

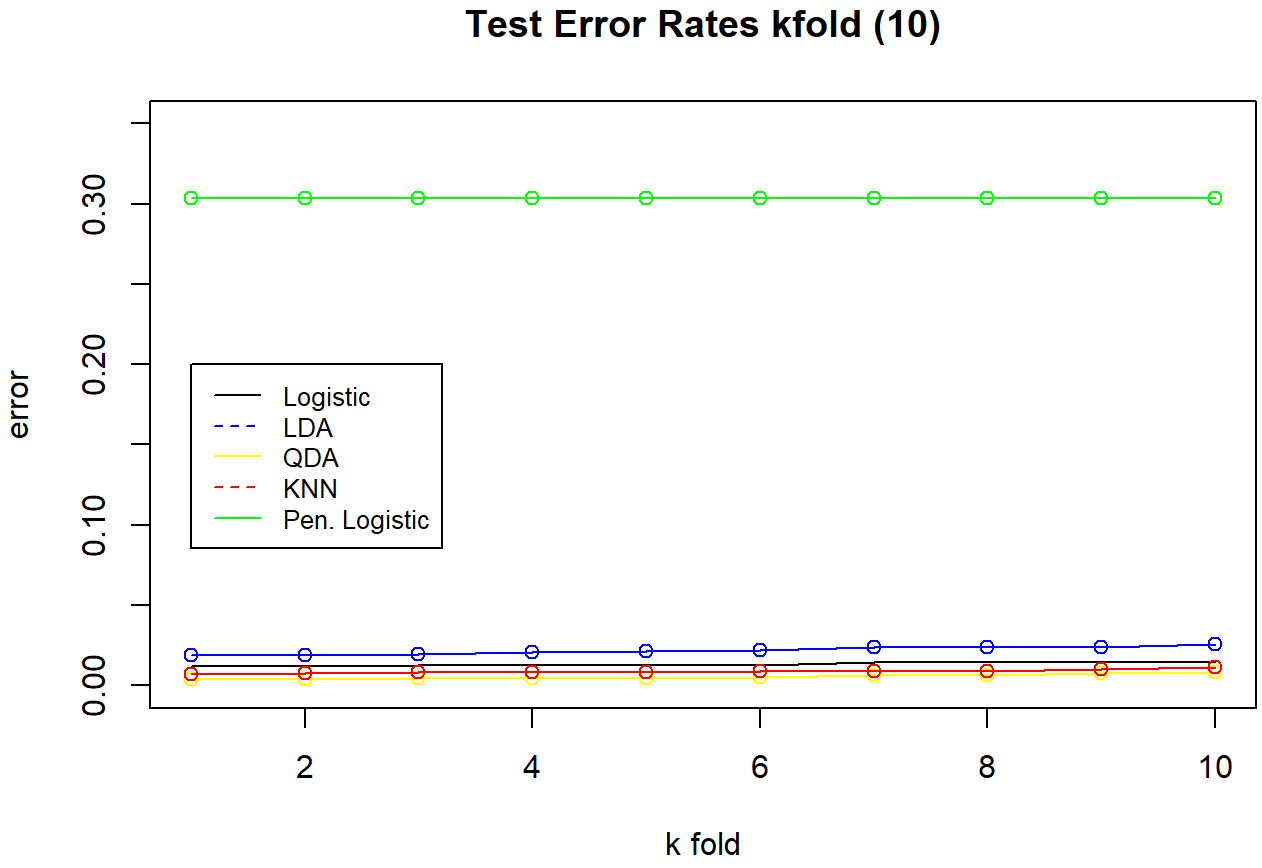


Results (Cross Validation showing k fold MSE Comparison)

```r
plot(sort(logistic.error), type = "l", ylim=c(0,.03), main = "Test Error Rates kfold (10)
- WO PLR", ylab="error", xlab="k fold")
points(sort(lda.error), col = "blue")
lines(sort(lda.error), col = "blue")

points(sort(qda.error),col = "yellow")
lines(sort(qda.error),col = "yellow")

points(sort(knn.error), col = "red")
lines(sort(knn.error), col = "red")

#points(sort(plr.error), col = "green")
#lines(sort(plr.error), col = "green")

legend(1,.03, legend=c("Logistic","LDA", "QDA", "KNN", "Pen. Logistic (out of range)"),col
=c("black","blue","yellow","red", "green"), lty=1:2, cex=0.8)
```



```
plot(sort(logistic.error), type = "l", ylim=c(0,.35), main = "Test Error Rates kfold (10)"
        , ylab="error", xlab="k fold")
points(sort(lda.error), col = "blue")
lines(sort(lda.error), col = "blue")

points(sort(qda.error),col = "yellow")
lines(sort(qda.error),col = "yellow")

points(sort(knn.error), col = "red")
lines(sort(knn.error), col = "red")

points(sort(plr.error), col = "green")
lines(sort(plr.error), col = "green")

legend(1,.20, legend=c("Logistic","LDA", "QDA", "KNN", "Pen. Logistic"),col=c("black","blu
        e","yellow","red","green"), lty=1:2, cex=0.8)
```



4 Performance Table

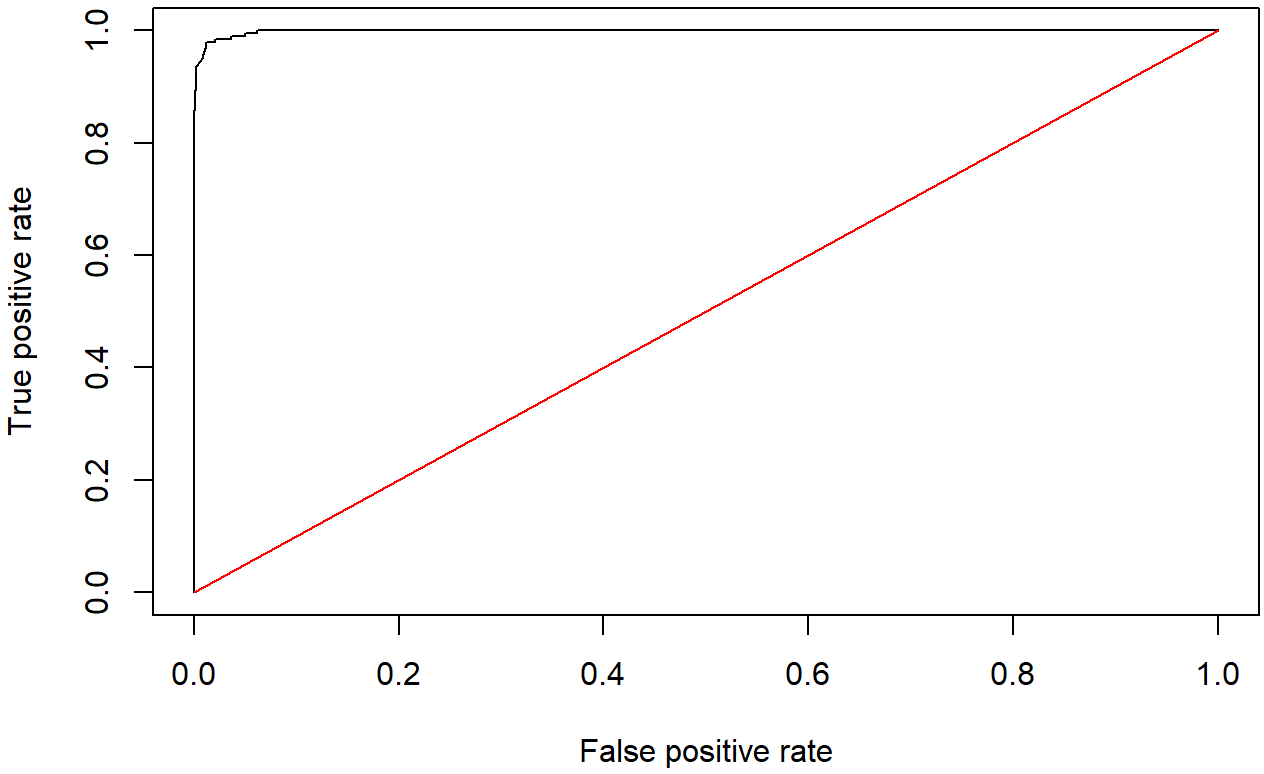
```
pt <- setNames(data.frame(matrix(ncol = 8, nrow = 5)), c("Model", "Tuning", "AUROC" , "Thr
         eshold" , "Accuracy", "TPR", "FPR" , "Precision"))

pt[1,1] = "Logistic"
pt[2,1] = "LDA"
pt[3,1] = "QDA"
pt[4,1] = "KNN"
pt[5,1] = "PLR"
pt[6,1] = "R.Forest"
pt[7,1] = "SVM"

pt[1,2] = "   *   "  #tuning
pt[1,3] = formatC(lm.auc)           #AUC
pt[1,4] = formatC(lm.cutoff)        #threshold
pt[1,5] = formatC((lm.table[1,1]+lm.table[2,2]) / sum(lm.table)) #Accuracy
pt[1,6] = formatC((lm.table[2,2]) / sum(lm.table[2,]))          #TPR
pt[1,7] = formatC(lm.table[1,2] / sum(lm.table[1,]))            #FPR
pt[1,8] = formatC((lm.table[2,2])/ sum(lm.table[,2]))           #Precision


pt[2,2] = "   *   "  #tuning
pt[2,3] = formatC(lda.auc)           #AUC
pt[2,4] = formatC(lda.cutoff)        #threshold
pt[2,5] = formatC((lda.table[1,1]+lda.table[2,2]) / sum(lda.table)) #Accuracy
pt[2,6] = formatC((lda.table[2,2]) / sum(lda.table[2,]))          #TPR
pt[2,7] = formatC(lda.table[1,2] / sum(lda.table[1,]))            #FPR
pt[2,8] = formatC((lda.table[2,2])/ sum(lda.table[,2]))           #Precision



pt[3,2] = "   *   "  #tuning
pt[3,3] = formatC(qda.auc)           #AUC
pt[3,4] = formatC(qda.cutoff)        #threshold
pt[3,5] = formatC((qda.table[1,1]+qda.table[2,2]) / sum(qda.table)) #Accuracy
pt[3,6] = formatC((qda.table[2,2]) / sum(qda.table[2,]))          #TPR
pt[3,7] = formatC(qda.table[1,2] / sum(qda.table[1,]))            #FPR
pt[3,8] = formatC((qda.table[2,2])/ sum(qda.table[,2]))           #Precision




pt[4,2] = "k= 5    "  #tuning
pt[4,3] = formatC(knn.auc)           #AUC
pt[4,4] = formatC(.5)        #threshold
pt[4,5] = formatC((knn.table[1,1]+knn.table[2,2]) / sum(knn.table)) #Accuracy
pt[4,6] = formatC((knn.table[2,2]) / sum(knn.table[2,]))          #TPR
pt[4,7] = formatC(knn.table[1,2] / sum(knn.table[1,]))            #FPR
pt[4,8] = formatC((knn.table[2,2])/ sum(knn.table[,2]))           #Precision


pt[5,2] = formatC(cvfit$lambda.min)  #tuning
pt[5,3] = formatC(plr.auc)           #AUC
```

```
pt[5,4] = formatC(.5)        #threshold
pt[5,5] = formatC((plr.table[1,1]+plr.table[2,2]) / sum(plr.table)) #Accuracy
pt[5,6] = formatC((plr.table[2,2]) / sum(plr.table[2,]))          #TPR
pt[5,7] = formatC(plr.table[1,2] / sum(plr.table[1,]))            #FPR
pt[5,8] = formatC((plr.table[2,2])/ sum(plr.table[,2]))           #Precision



pt[6,2] = "mtry = 3 ntree=500"  #tuning
pt[6,3] = formatC(rf.auc)           #AUC
pt[6,4] = formatC(rf.cutoff)        #threshold
pt[6,5] = formatC((rf.table[1,1]+rf.table[2,2]) / sum(rf.table)) #Accuracy
pt[6,6] = formatC((rf.table[2,2]) / sum(rf.table[2,]))          #TPR
pt[6,7] = formatC(rf.table[1,2] / sum(rf.table[1,]))            #FPR
pt[6,8] = formatC((rf.table[2,2])/ sum(rf.table[,2]))           #Precision

pt[7,2] = "cost=1, gamma=2, kernel=radial"  #tuning
pt[7,3] = formatC(svm.auc)           #AUC
pt[7,4] = formatC(svm.cutoff)        #threshold
pt[7,5] = formatC((svm.table[1,1]+svm.table[2,2]) / sum(svm.table)) #Accuracy
pt[7,6] = formatC((svm.table[2,2]) / sum(svm.table[2,]))          #TPR
pt[7,7] = formatC(svm.table[1,2] / sum(svm.table[1,]))            #FPR
pt[7,8] = formatC((svm.table[2,2])/ sum(svm.table[,2]))           #Precision


#kable(pt[1:5,], width="20cm" , caption = "Performance Table")
```

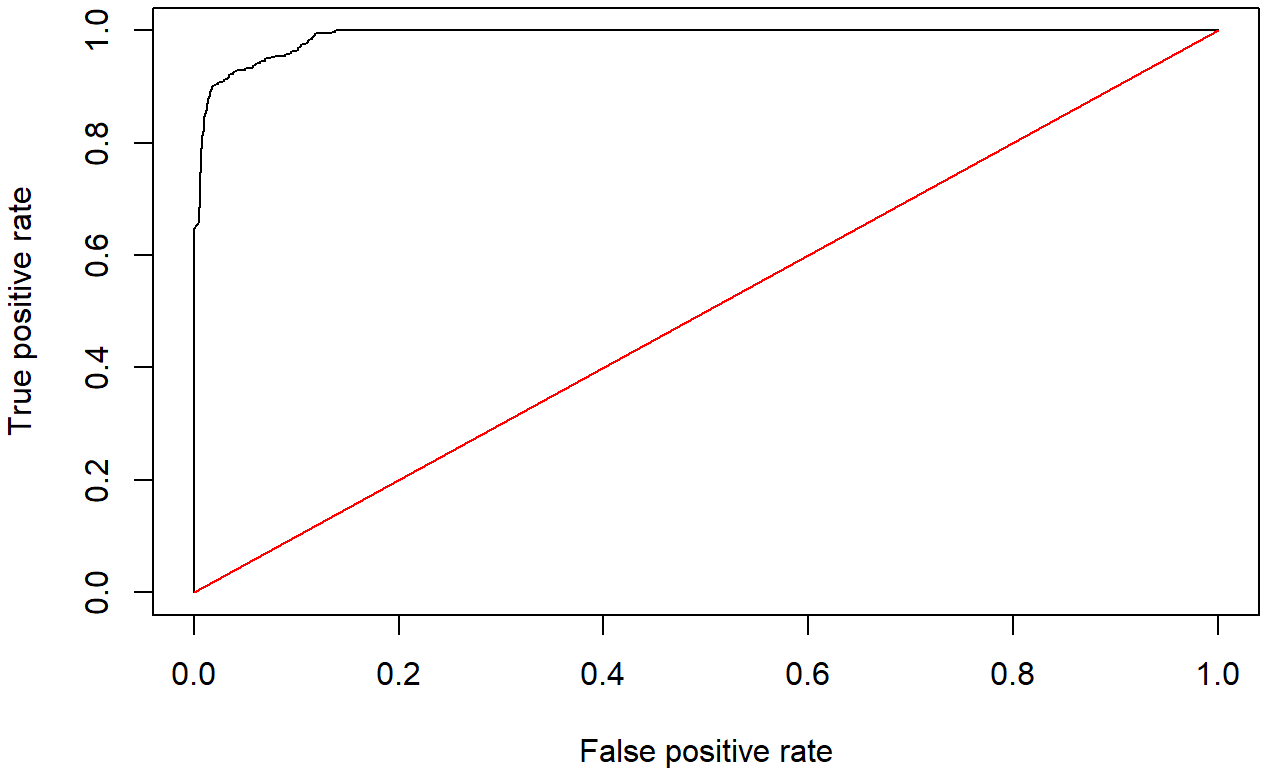
4.1 ROC CURVES

The ROC Curves are built from out-of sample data (testing data).

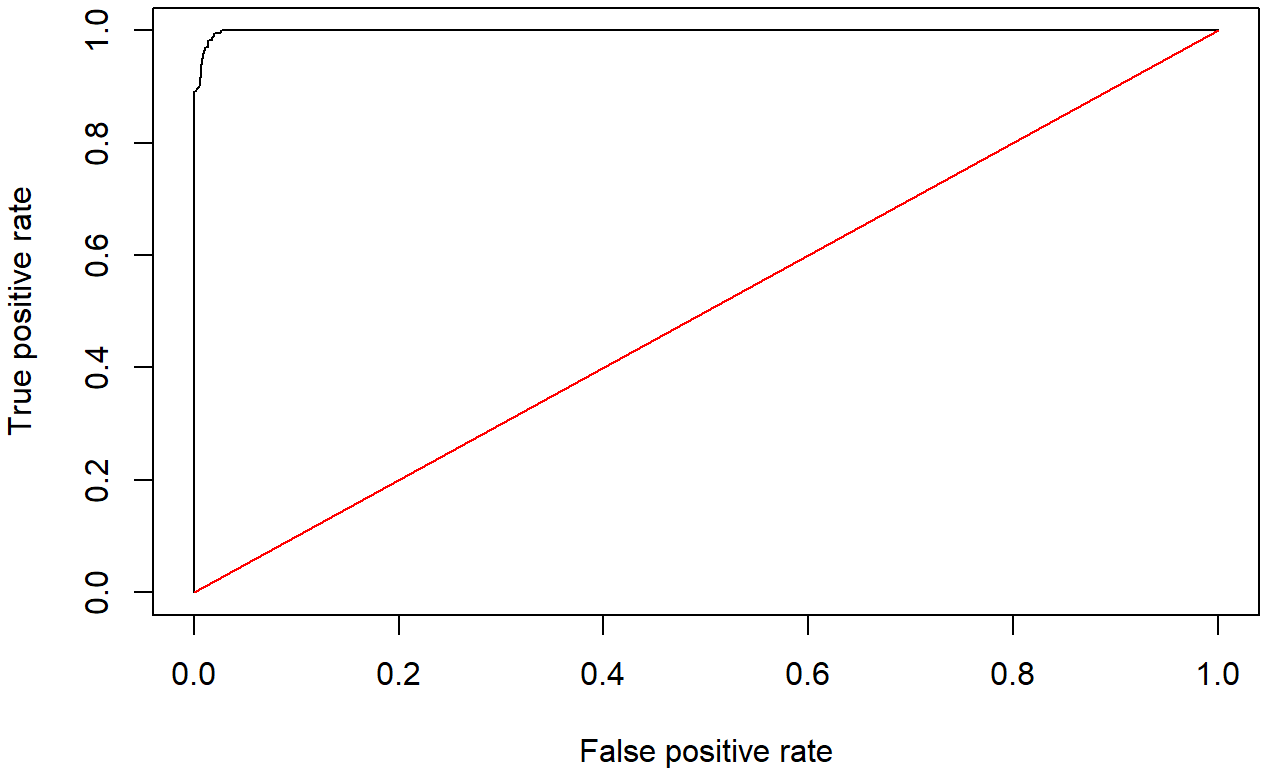
```
library(ROCR)

#Logistic ROC
c_rate<-prediction(p_hat, haiti$Class.Dummy[test])
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from Logistic Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```



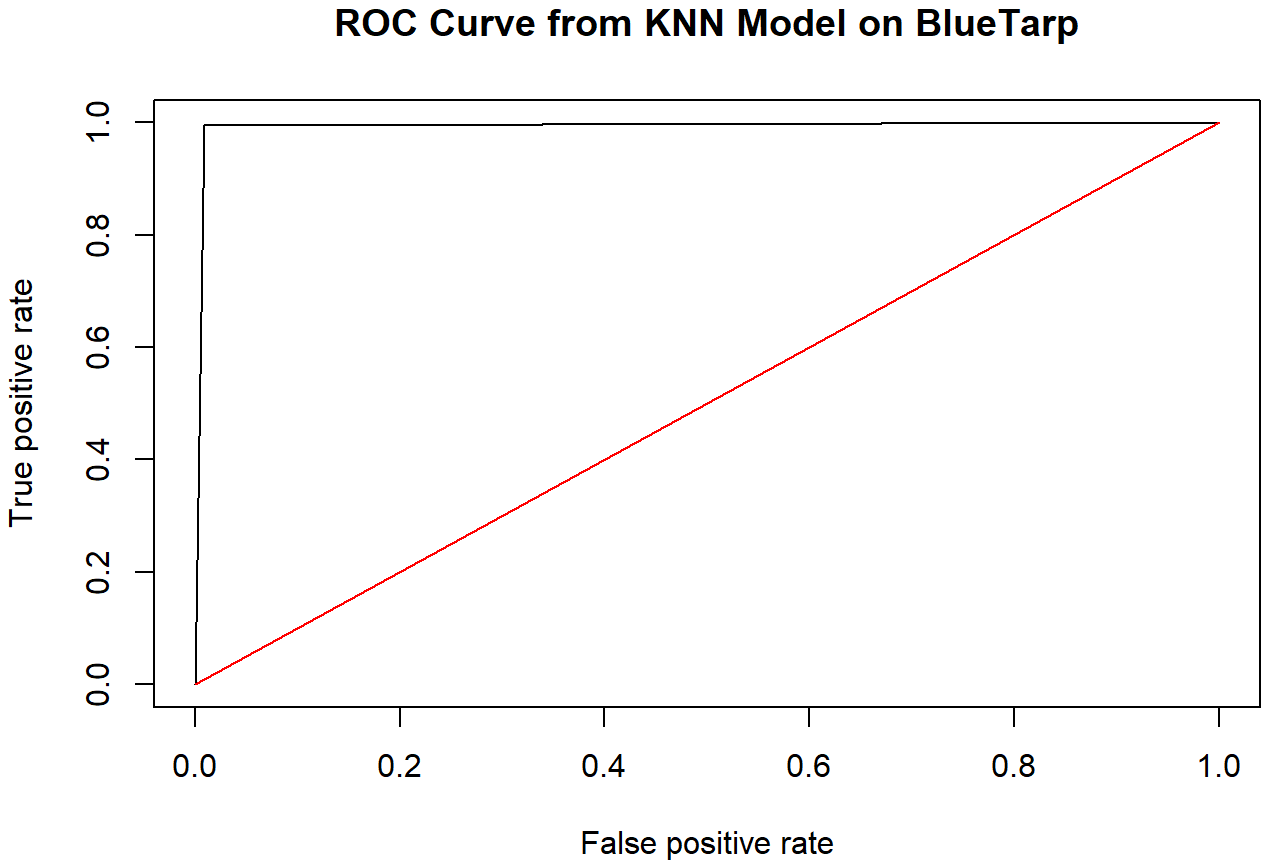

```
#LDA ROC
#Apply to test data

lda.pred = predict(lda.fit, haiti[haiti.test,])

#Convert posteriors to dataframe
lda.pred.post <-as.data.frame(lda.pred$posterior)

#Classification table for ROC
lda.btarp<-prediction(lda.pred.post[,2], as.data.frame(haiti$Class.Dummy[haiti.test]))
roc_data<-performance(lda.btarp, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from LDA Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```



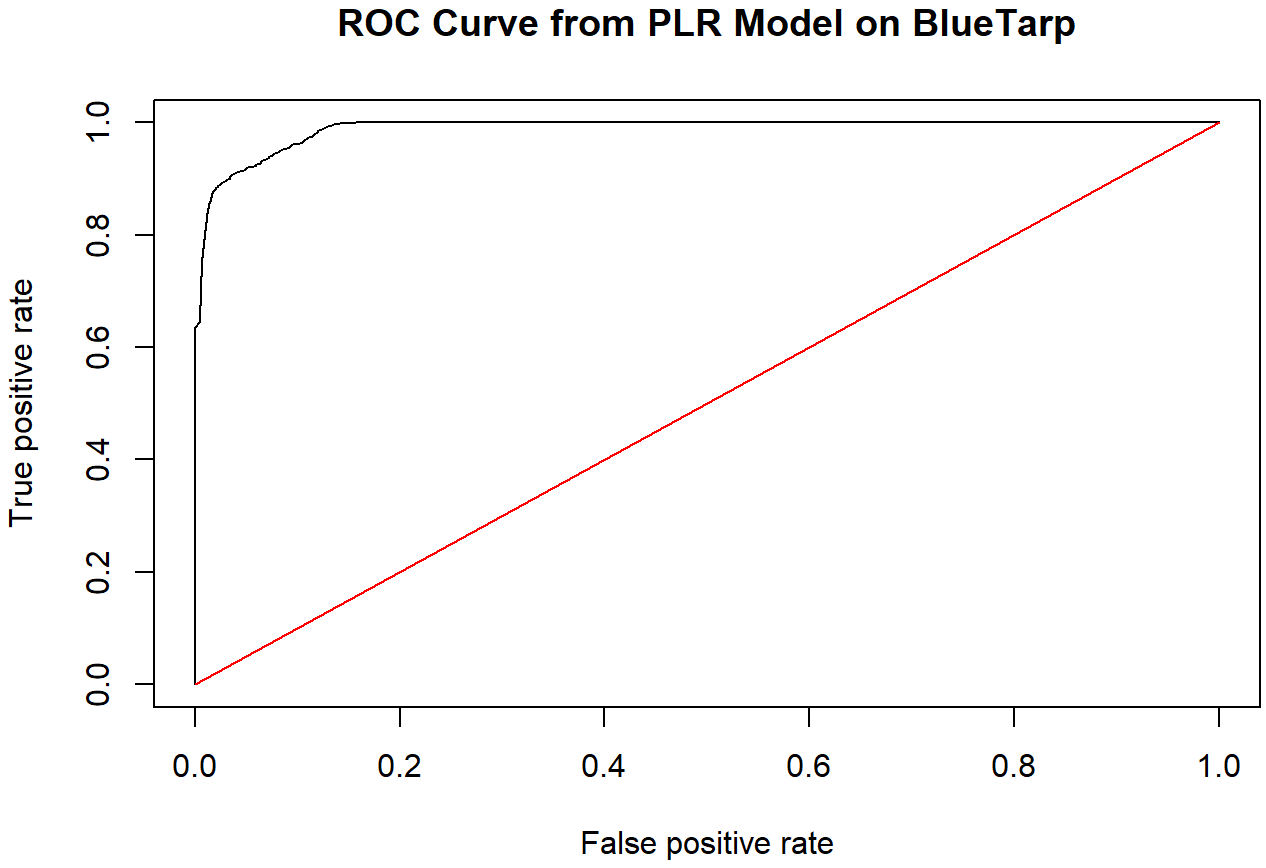

```
#QDA ROC
#Apply to test data

qda.class = predict(qda.fit,haiti[haiti.test,] )$class #predictions of BlueTarp on test da
         ta
qda.pred = predict(qda.fit,haiti[haiti.test,] )

#Classification table for ROC
qda.btarp<-prediction(qda.pred$posterior[,2], as.data.frame(haiti$Class.Dummy[haiti.tes
         t]))

roc_data<-performance(qda.btarp, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from QDA Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```



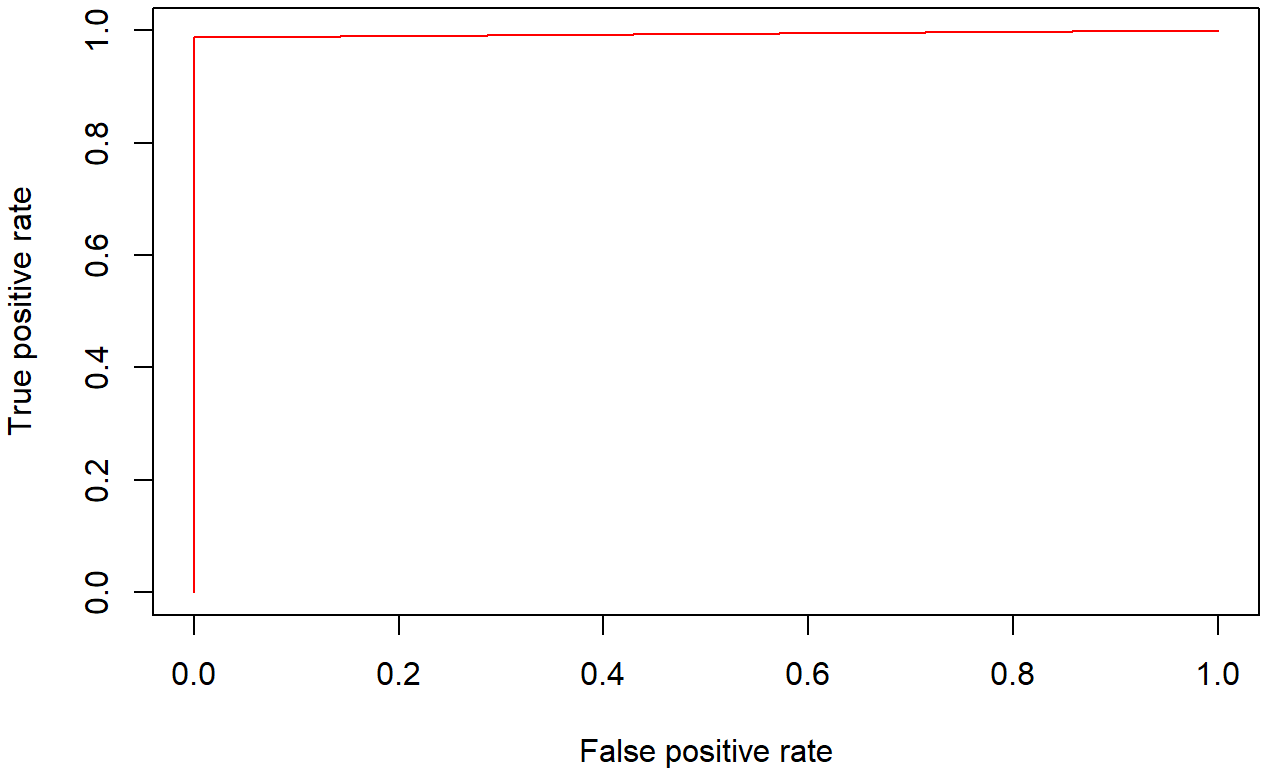

```
#KNN Classification table for ROC
#Classification table for ROC
c_rate<-prediction(knn.pred.class, test.Class.Dummy )
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from KNN Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```



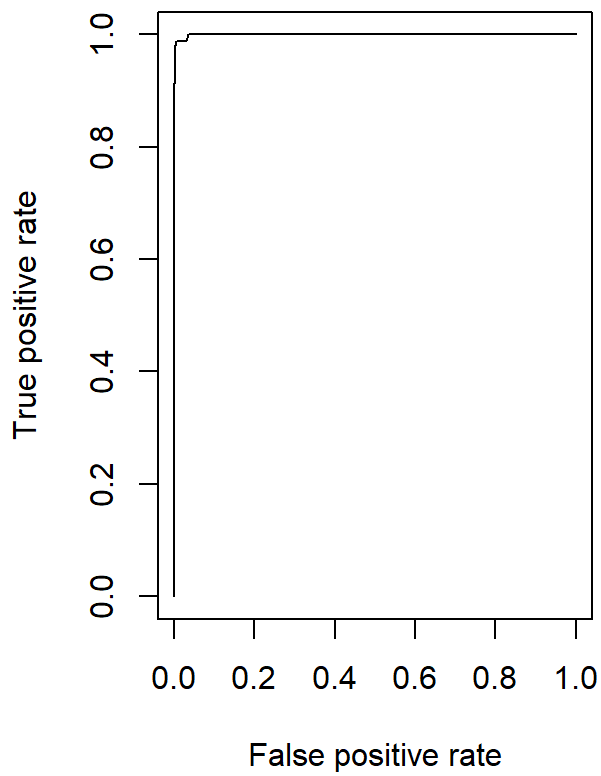

```
#PLR ROC
c_rate<-prediction(predict.BlueTarp, train.Class.Dummy)
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from PLR Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Classification table for ROC Random F.
predictions <- as.numeric(predict(bag.haiti, haiti, type="response"))
pred <- prediction(predictions, haiti$Class.Dummy)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, col=rainbow(10), main="ROC Curve from Random Forest Model on BlueTarp")
```

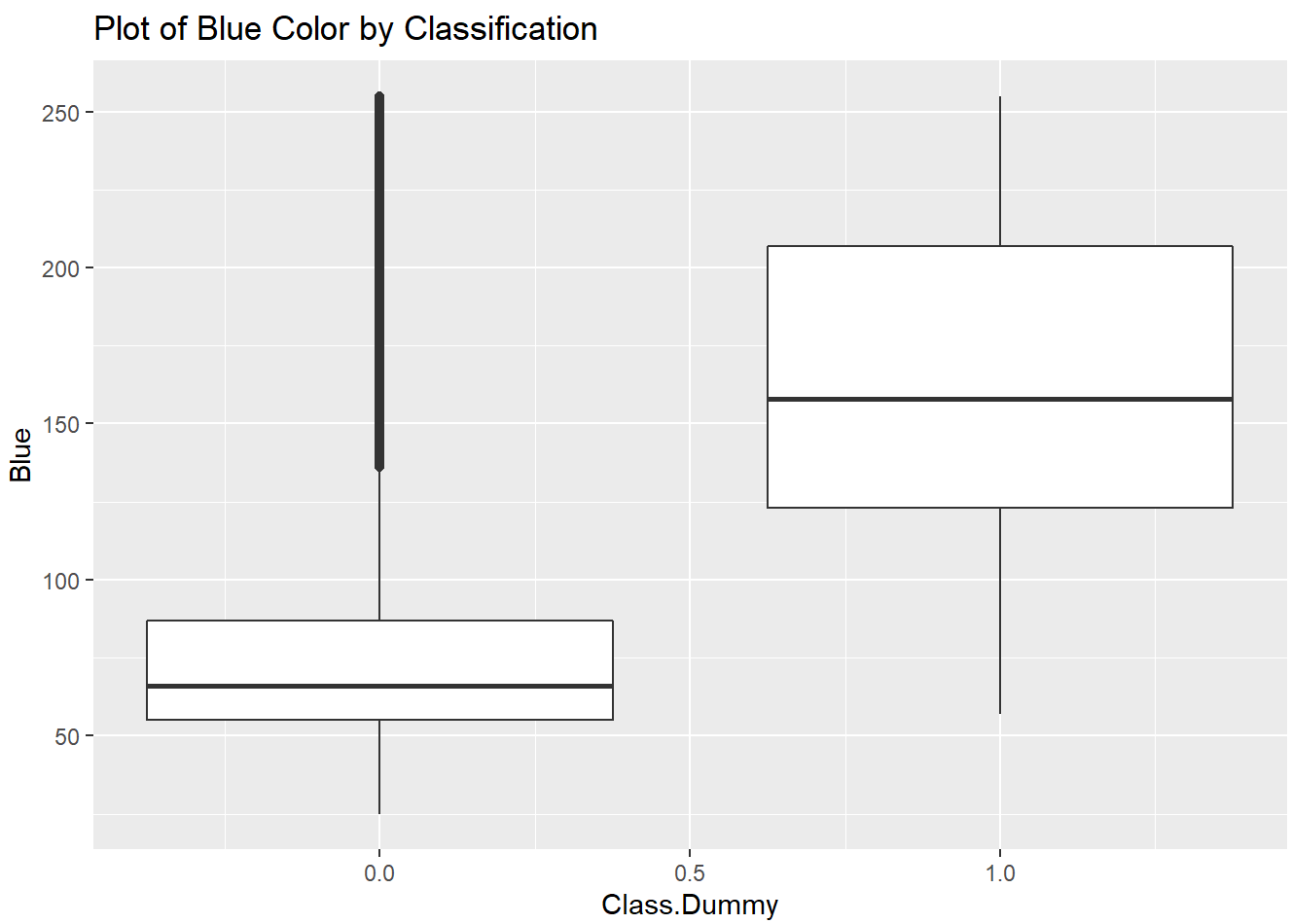



```
#ROC Curves for SVM
library (ROCR)
rocplot=function(pred.te,truth, ...) {
  predob = prediction(pred.te,truth)
  perf = performance(predob, "tpr" ,"fpr") #ignore the axes labels!! had to switch fpr and
         tpr to make plot correct
  plot(perf,...)}


par(mfrow=c(1,2)) #create one row two columns of figures to plot

rocplot(fitted, haiti[,"Class.Dummy"],main="ROC Curve from SVM Model on BlueTarp")
```

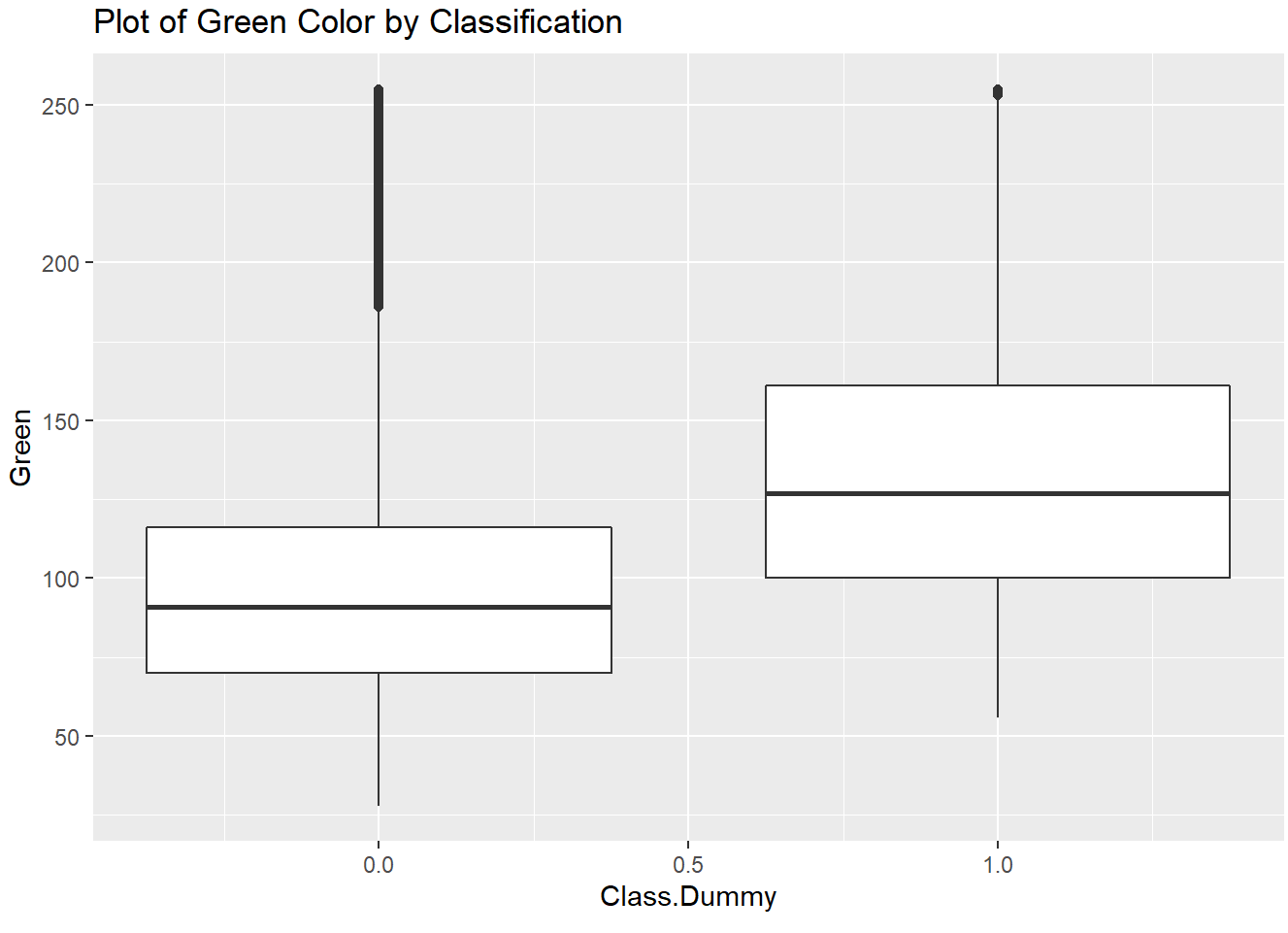


5 Read in Hold Out Data

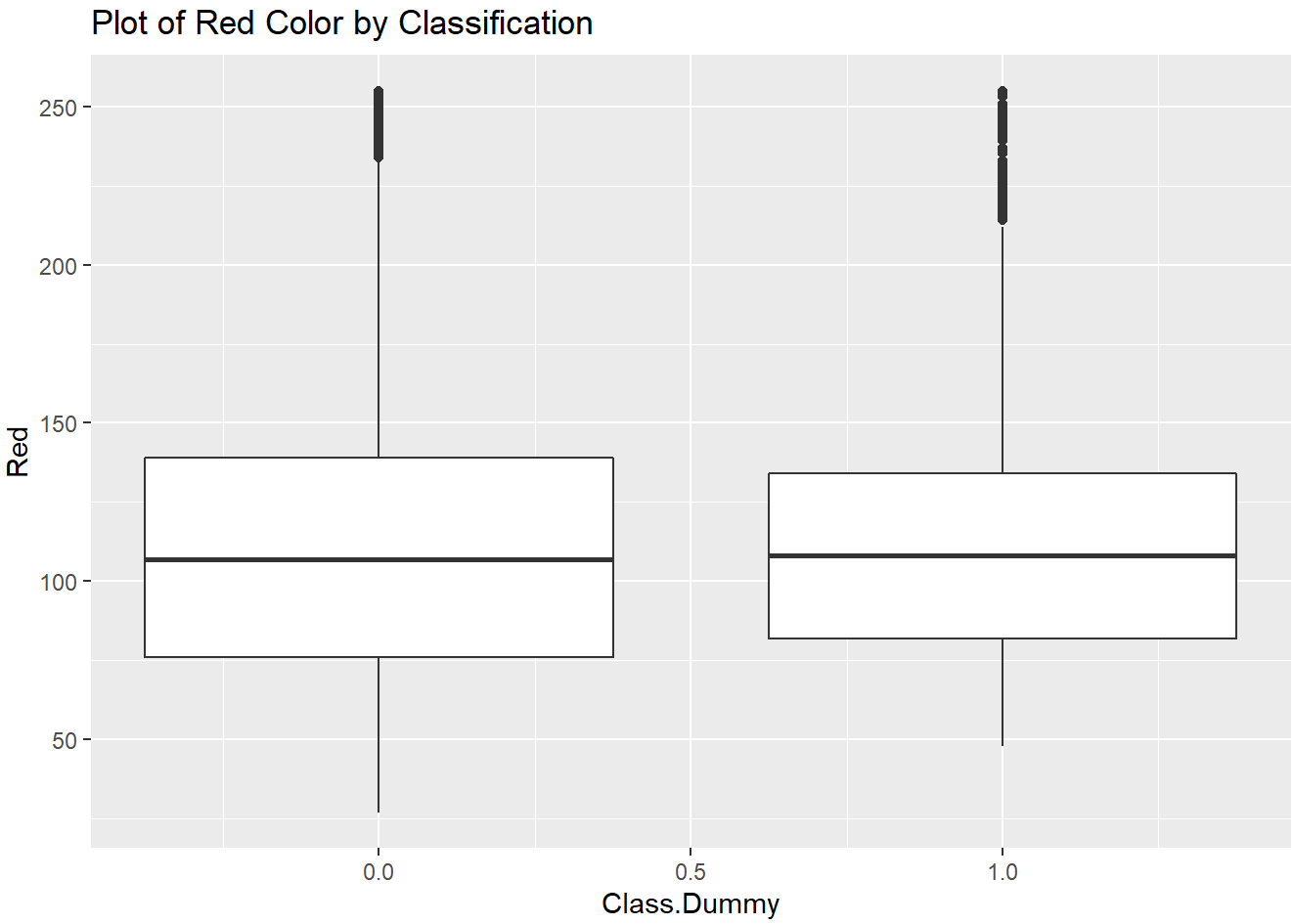
```
library(reader)
library(readr)

holdout1 <- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/ort
        hovnir057_ROI_NON_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))
holdout2<- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/orth
        ovnir067_ROI_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))
#holdout22<- read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/orthovnir
        067_ROI_Blue_Tarps_data.txt") #no skip, just B1, B2, B3 may be duplicate?
holdout3 <- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/ort
        hovnir067_ROI_NOT_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))
holdout4 <- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/ort
        hovnir069_ROI_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))
holdout5 <- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/ort
        hovnir069_ROI_NOT_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))
holdout6 <- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/ort
        hovnir078_ROI_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))
holdout7 <- subset(read.table("C:/Users/ajoss/Documents/Data Mining SYS6018 Spring2021/ort
        hovnir078_ROI_NON_Blue_Tarps.txt", skip=8),select=-c(V1,V2,V3,V4,V5,V6,V7))



Class.Dummy <- 1;
holdout.bluetarp <-cbind(Class.Dummy,rbind(holdout2,holdout4, holdout6))

Class.Dummy <- 0
holdout.NoTarp <-cbind(Class.Dummy,rbind(holdout1,holdout3,holdout5, holdout7))

holdout.all <- rbind(holdout.bluetarp,holdout.NoTarp)
names(holdout.all) <- c('Class.Dummy','Red','Green','Blue')

library(ggplot2)
ggplot(holdout.all, aes(x=Class.Dummy, y=Blue, group=Class.Dummy)) +
  geom_boxplot() +  ggtitle("Plot of Blue Color by Classification")
```




```
ggplot(holdout.all, aes(x=Class.Dummy, y=Green, group=Class.Dummy)) +
  geom_boxplot() +  ggtitle("Plot of Green Color by Classification")
```



```
ggplot(holdout.all, aes(x=Class.Dummy, y=Red, group=Class.Dummy)) +
  geom_boxplot() +  ggtitle("Plot of Red Color by Classification")
```



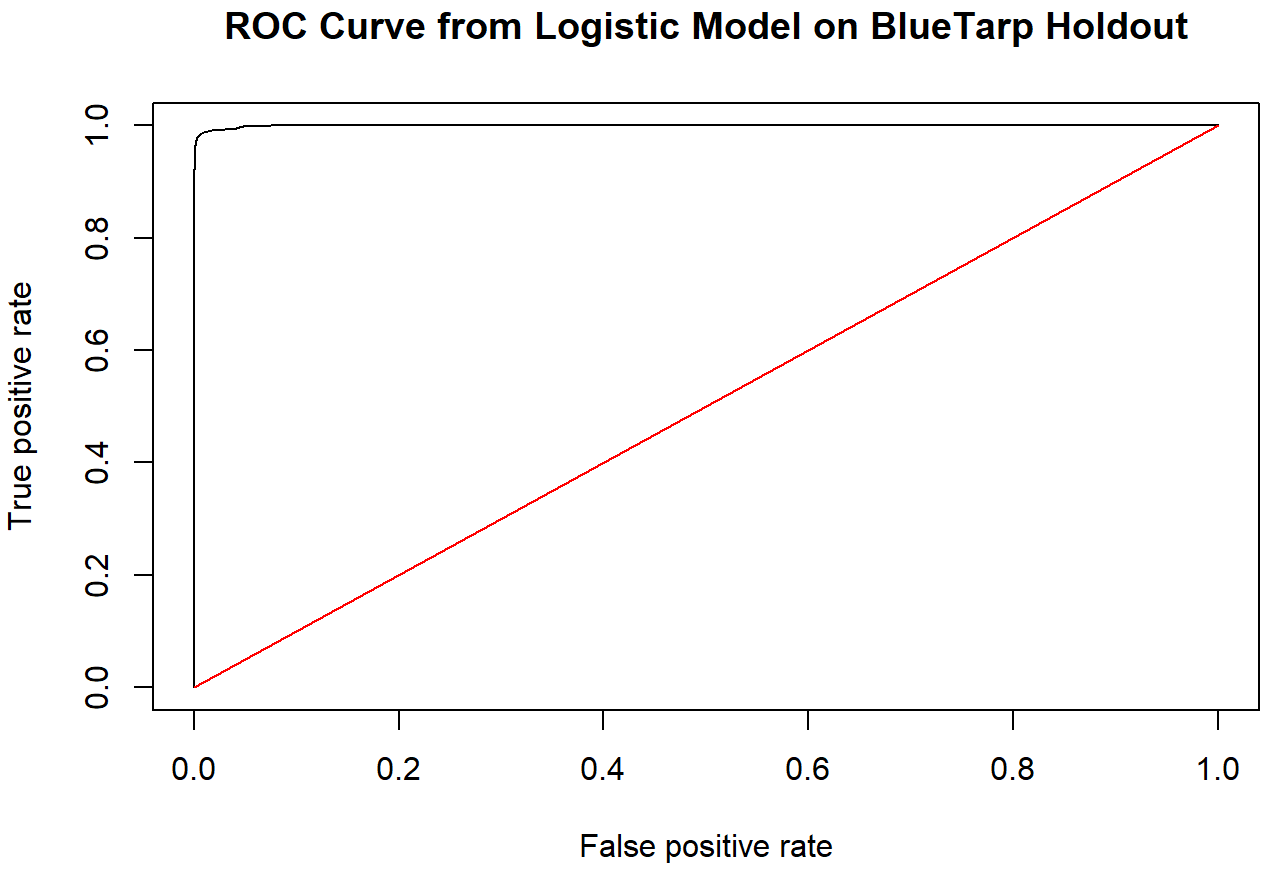
6 Apply Methods to Holdout Data

6.1 Logistic to HOLDOUT DATA

```
#full model
glm.predict<-glm(Class.Dummy~Red+Green+Blue, data=haiti, family=binomial)
summary(glm.predict)
```

```
##
## Call:
## glm(formula = Class.Dummy ~ Red + Green + Blue, family = binomial,
##     data = haiti)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4266  -0.0205  -0.0015   0.0000   3.7911
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20984    0.18455   1.137    0.256
## Red         -0.26031    0.01262 -20.632   <2e-16 ***
## Green       -0.21831    0.01330 -16.416   <2e-16 ***
## Blue         0.47241    0.01548  30.511   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 17901.6  on 63240  degrees of freedom
## Residual deviance:  1769.5  on 63237  degrees of freedom
## AIC: 1777.5
##
## Number of Fisher Scoring iterations: 12
```

```
p_hat.ho = predict(glm.predict, holdout.all, type = "response")

#Classification table for ROC
c_rate<-prediction(p_hat.ho, holdout.all$Class.Dummy)
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from Logistic Model on BlueTarp Holdout")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC
auc<-performance(c_rate, measure = "auc")
lm.auc.ho <-auc@y.values[[1]]

##confusion matrix
lm.cutoff.ho = .034
lm.table.ho <- table(holdout.all$Class.Dummy, p_hat.ho>lm.cutoff.ho)
lm.table.ho
```

```
##
##       FALSE    TRUE
##   0 1769974  219723
##   1       0   14480
```

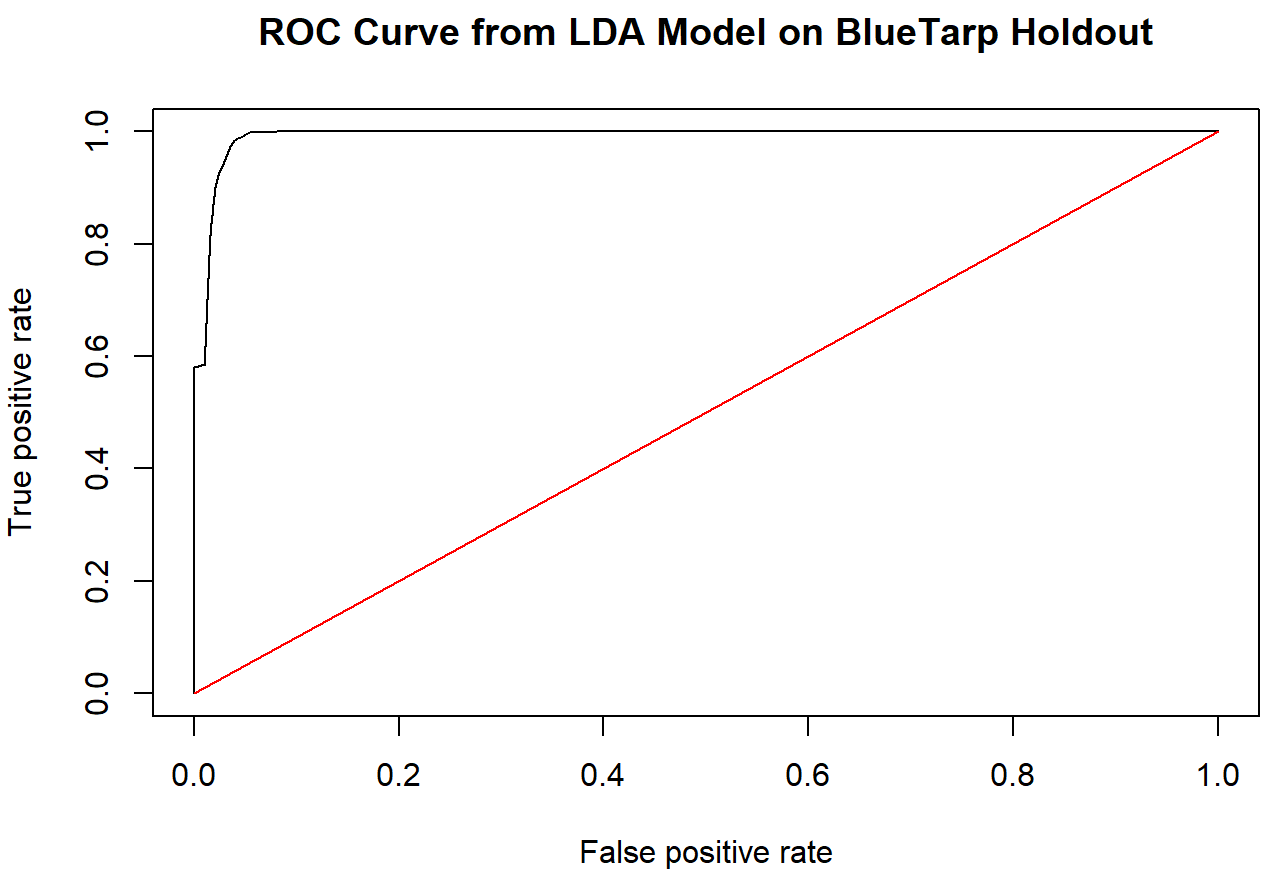
```
lm.applied.ho <- cbind(holdout.all$Class.Dummy,p_hat.ho>lm.cutoff.ho)
```

6.1.1 Logistic holdout observations

I correctly predicted all of the blue tarps as blue tarps. There is a large amount of obs identified as having tarps that do not, but I have 0 identified as not having a tarp that actually do. And would prefer to error on this side.

6.2 LDA to HOLDOUT DATA

```
#Apply to test data

lda.fit=lda(Class.Dummy~Red+Green+Blue, data=haiti)
lda.pred.ho = predict(lda.fit, holdout.all)

#Convert posterirors to dataframe
lda.pred.post.ho <-as.data.frame(lda.pred.ho$posterior)


 # Predictions

  lda.class.ho = lda.pred.ho$class #predictions of BlueTarp on holdout data
  lda.cutoff = .015
  lda.class.adj.ho <- ifelse(lda.pred.ho$posterior[, 2] > lda.cutoff, "TRUE", "FALSE")  #t
         ried between .01 & .035
  lda.class.adj.ho <-lda.class.adj.ho == "TRUE"


#Classification table for ROC
lda.btarp<-prediction(lda.pred.post.ho[,2], as.data.frame(holdout.all$Class.Dummy))
roc_data<-performance(lda.btarp, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from LDA Model on BlueTarp Holdout")
lines( x = c(0,1), y = c(0,1),col="red")
```



```
#Find AUC (higher is better)
auc<-performance(lda.btarp, measure = "auc")
lda.auc.ho = auc@y.values[[1]]

##confusion matrix
lda.table.ho = table(holdout.all$Class.Dummy,lda.class.adj.ho)
lda.table.ho
```

```
##    lda.class.adj.ho
##       FALSE    TRUE
##   0 1930733   58964
##   1     786   13694
```

6.2.1 LDA holdout observations

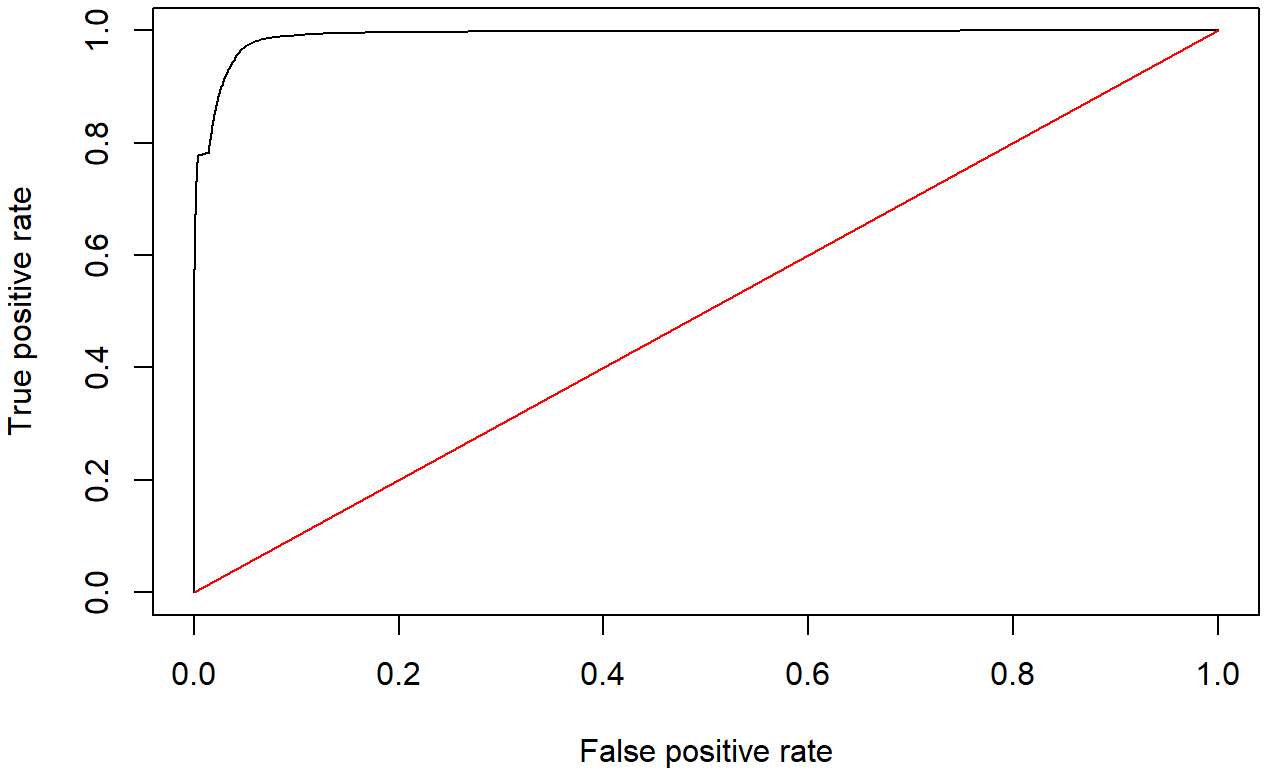
786 obs were incorrectly identified as not having tarps

6.3 QDA to Holdout DATA

```
qda.fit<- qda(Class.Dummy ~ Red + Green + Blue, data=haiti)
qda.fit
```

```
## Call:
## qda(Class.Dummy ~ Red + Green + Blue, data = haiti)
##
## Prior probabilities of groups:
##          0          1
## 0.96802707 0.03197293
##
## Group means:
##        Red    Green     Blue
## 0 162.7604 152.5808 122.4993
## 1 169.6627 186.4149 205.0371
```

```
qda.class.ho = predict(qda.fit,holdout.all )$class #predictions of BlueTarp on test data
qda.pred.ho = predict(qda.fit,holdout.all )

qda.cutoff.ho=.5

#Classification table for ROC
qda.btarp<-prediction(qda.pred.ho$posterior[,2], as.data.frame(holdout.all$Class.Dummy))

roc_data<-performance(qda.btarp, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from QDA Model on BlueTarp")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC (higher is better)
auc<-performance(qda.btarp, measure = "auc")
qda.auc.ho = auc@y.values[[1]]

##confusion matrix
qda.table.ho = table(holdout.all$Class.Dummy,qda.class.ho)
qda.table.ho
```

```
##    qda.class.ho
##           0       1
##   0 1986046    3651
##   1    4422   10058
```

6.4 KNN to HOLDOUT DATA

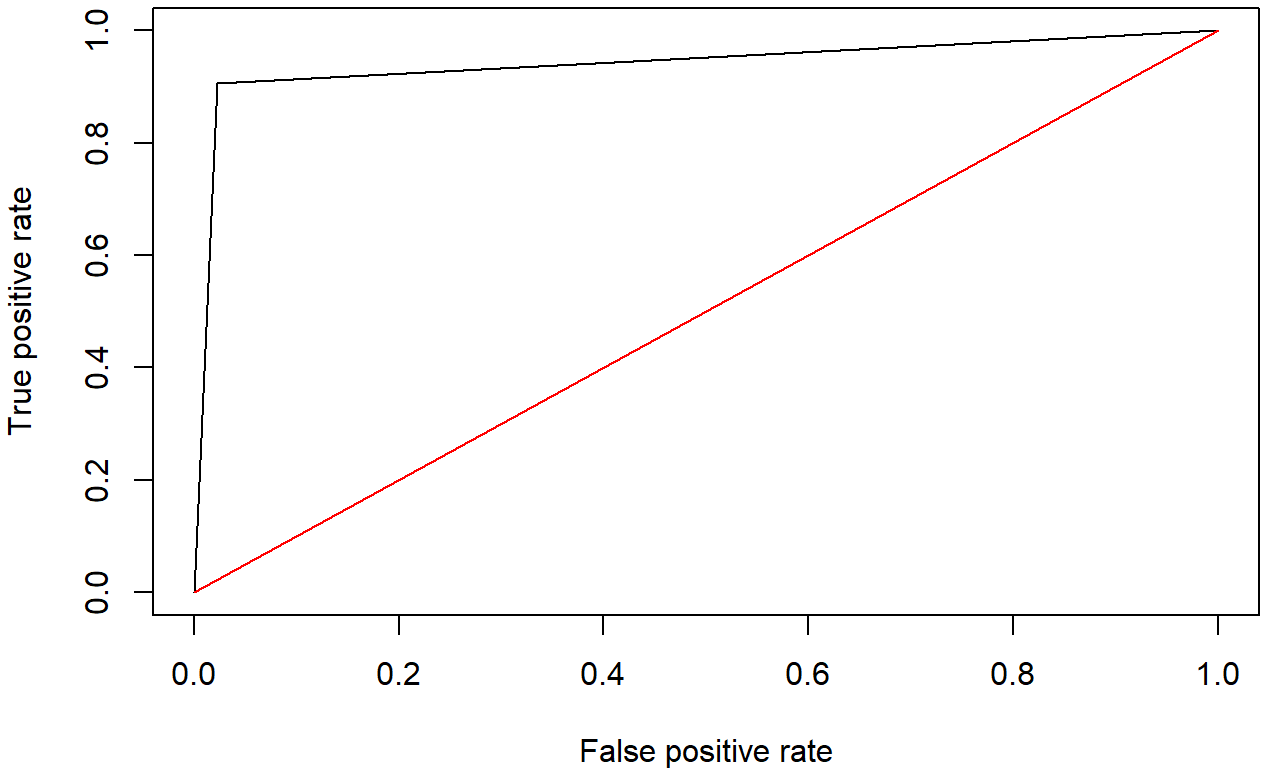
```
library(ROCR)

library(ISLR)
library(caret)

knn_mod = knn3(Class.Dummy~Red+Green+Blue, data=haiti, k = 5) #knn model
knn.pred.ho <- predict(knn_mod, holdout.all, type = "prob") #predict onto holdout


knn.prob.ho=knn.pred.ho[,2]
head(knn.prob.ho)
```

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

```
knn.pred.class <- knn.prob.ho > 0.05
knn.pred.class <- ifelse(knn.prob.ho > 0.05,1,0)



# Test Error

knn.table.ho = table(holdout.all$Class.Dummy,knn.pred.class)
knn.table.ho
```

```
##    knn.pred.class
##           0       1
##   0 1943847   45850
##   1    1365   13115
```

```
knn.error.ho = mean(Class.Dummy != knn.pred.class)



#Classification table for ROC
c_rate<-prediction(knn.pred.class, holdout.all$Class.Dummy)
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from KNN Model on BlueTarp Holdout")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC
auc<-performance(c_rate, measure = "auc")
knn.auc.ho <-auc@y.values[[1]]
```

6.5 PLR to HOLDOUT DATA

```
haiti.matrix <- data.matrix(haiti,rownames.force = NA)
holdout.all.matrix <- data.matrix(holdout.all, rownames.force = NA)


# Fitting the model & Predictions
fit <- glmnet(haiti.matrix[,2:4],as.vector(haiti$Class.Dummy))
cvfit <- cv.glmnet(haiti.matrix[ ,2:4],as.vector(haiti$Class.Dummy),nfolds=10)
predict.BlueTarp <- predict(fit, newx=holdout.all.matrix[,2:4],  s = cvfit$lambda.min)


# Test Error

summary(predict.BlueTarp, mean()) #to get cutoff  (too low)
```

```
##        1
##  Min.   :-0.41677
##  1st Qu.:-0.08036
##  Median :-0.03100
##  Mean   :-0.01498
##  3rd Qu.: 0.03158
##  Max.   : 0.84189
```

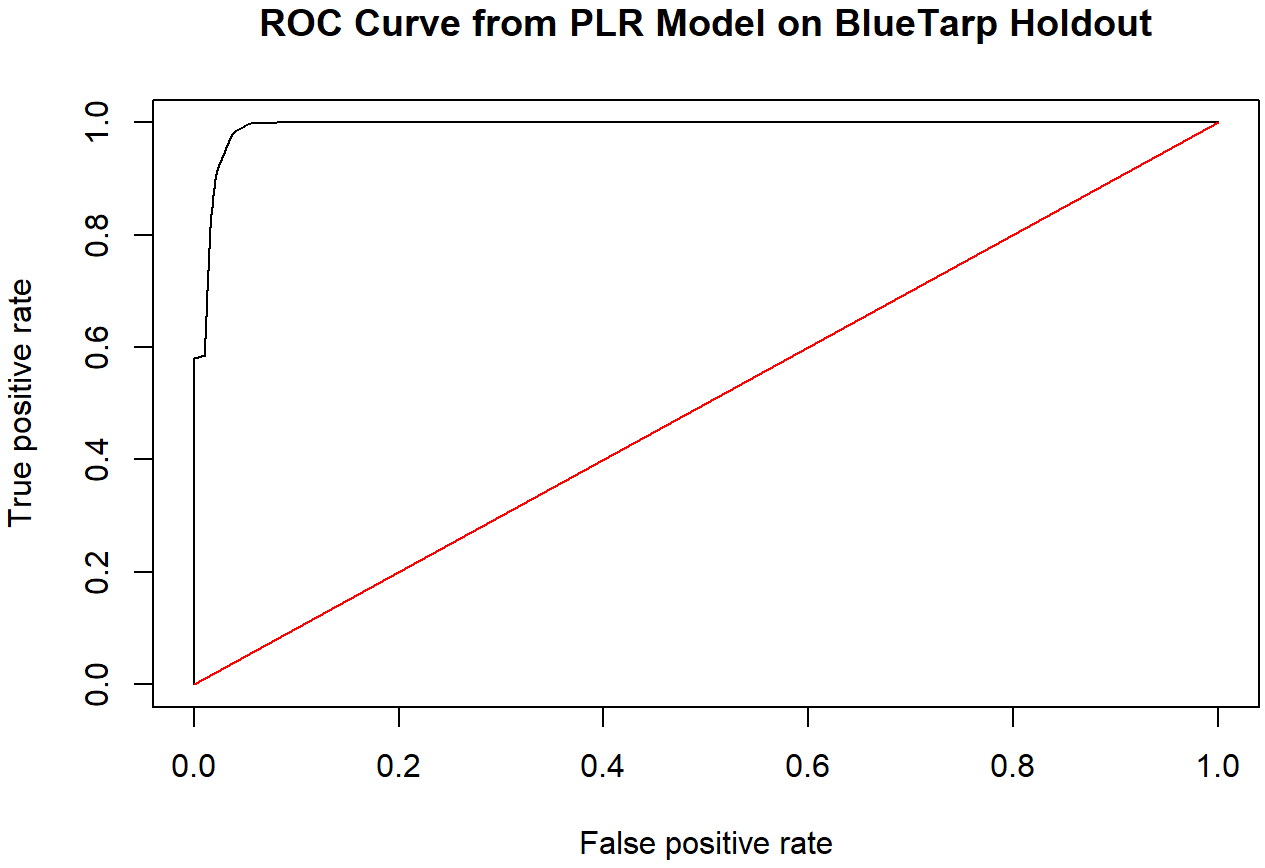
```
plr.cutoff = .09
classify.BlueTarp <- ifelse(predict.BlueTarp > plr.cutoff, 1, 0)

plr.table.ho = table(holdout.all$Class.Dummy,classify.BlueTarp)
plr.table.ho
```

```
##    classify.BlueTarp
##           0       1
##   0 1813057  176640
##   1       1   14479
```

```
plr.error.ho = mean(holdout.all$Class.Dummy != classify.BlueTarp)
plr.error.ho
```

```
## [1] 0.08813643
```

```
#Classification table for ROC
c_rate<-prediction(predict.BlueTarp, holdout.all$Class.Dummy)
roc_data<-performance(c_rate, measure="tpr" , x.measure="fpr")
plot(roc_data, main="ROC Curve from PLR Model on BlueTarp Holdout")
lines( x = c(0,1), y = c(0,1),col="red")
```




```
#Find AUC
auc<-performance(c_rate, measure = "auc")
plr.auc.ho <-auc@y.values[[1]]
```

6.6 RANDOM FOREST TO HOLDOUT DATA

```
library(randomForest)
library(pROC)
#https://stackoverflow.com/questions/46124424/how-can-i-draw-a-roc-curve-for-a-randomforest-model-with-three-classes-in-r

haiti.Class.Dummy <- as.factor(haiti$Class.Dummy)

bag.haiti <- randomForest(haiti.Class.Dummy~Red+Red^2+Green+Green^2+Blue+Blue^2+Blue^3,data=haiti, mtry=3,importance=TRUE, ntrees=500)

bag.haiti
```

```
##
## Call:
##  randomForest(formula = haiti.Class.Dummy ~ Red + Red^2 + Green +      Green^2 + Blue +
Blue^2 + Blue^3, data = haiti, mtry = 3,      importance = TRUE, ntrees = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.31%
## Confusion matrix:
##       0    1 class.error
## 0 61135   84 0.001372123
## 1   115 1907 0.056874382
```

```
yhat.bag <-predict(bag.haiti, newdata=holdout.all)
yhat.class <-as.data.frame(yhat.bag)

rf.table.ho <-table(holdout.all$Class.Dummy,yhat.bag) #(actual, predicted)
rf.table.ho
```

```
##    yhat.bag
##           0       1
##   0 1978488   11209
##   1    4149   10331
```

```
rf.error = mean(holdout.all$Class.Dummy != yhat.class$yhat.bag)
rf.error
```

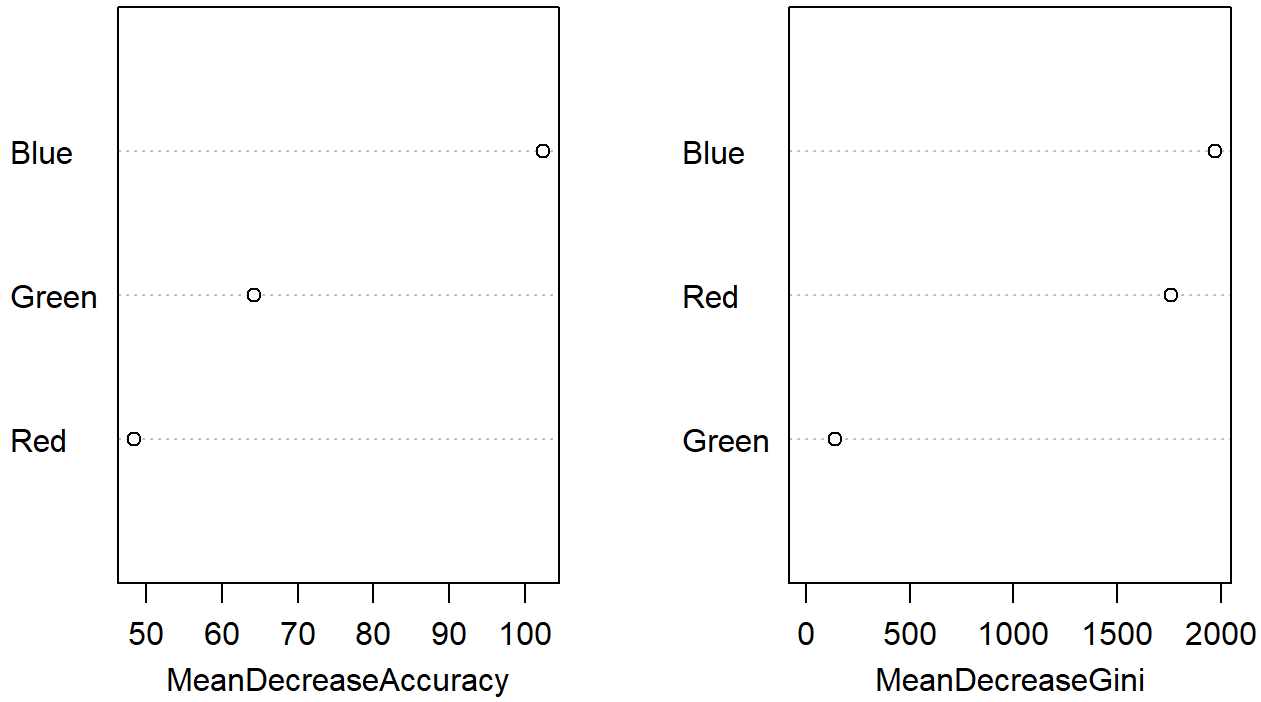
```
## [1] 0.007662996
```

```
importance(bag.haiti)
```

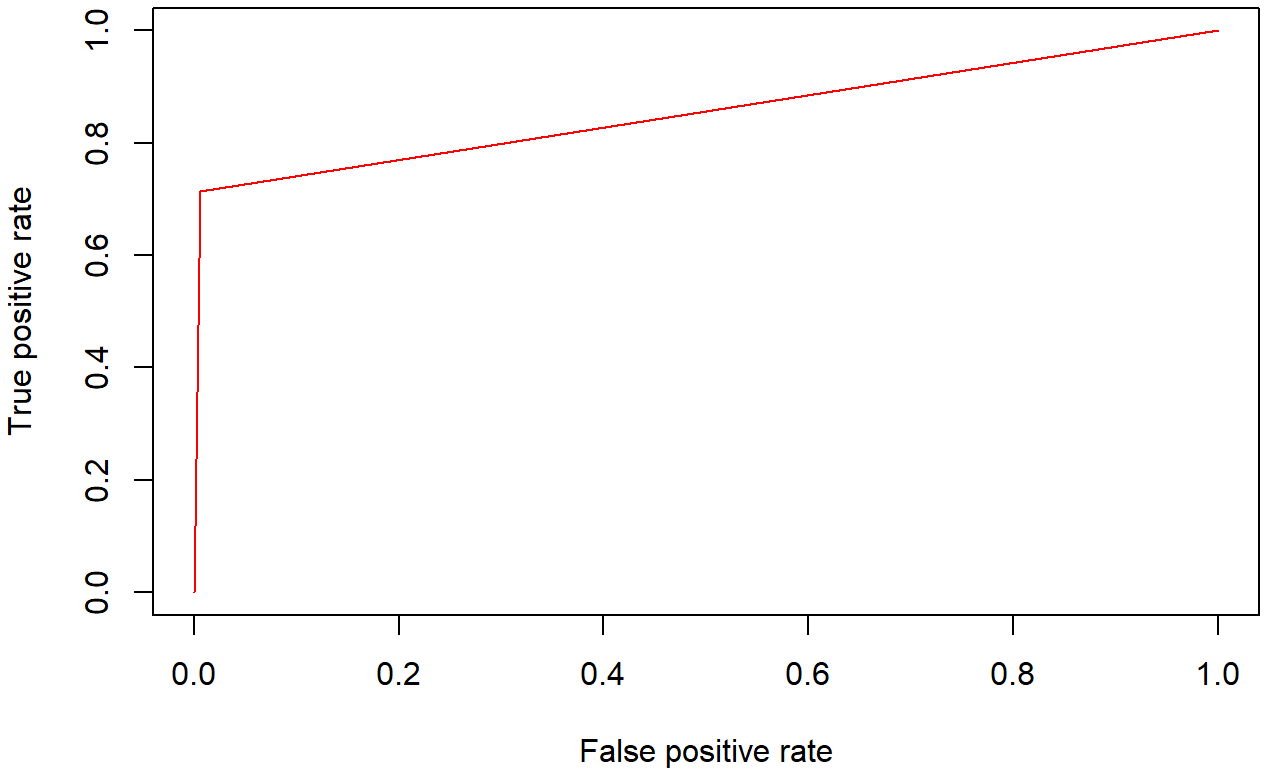
```
##               0        1 MeanDecreaseAccuracy MeanDecreaseGini
## Red   43.892245 143.3701             48.38116        1758.7930
## Green  7.401403  67.0844             64.25037         139.0859
## Blue  93.690949 690.4410            102.33803        1971.8431
```

```
varImpPlot(bag.haiti)
```

bag.haiti




```
#Classification table for ROC
predictions <- as.numeric(predict(bag.haiti, holdout.all, type="response"))
pred <- prediction(predictions, holdout.all$Class.Dummy)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, col=rainbow(10))
```




```
rf.auc.ho <- performance(pred,measure="auc")
rf.auc.ho <-auc@y.values[[1]]
```

6.7 SVM TO HOLDOUT DATA

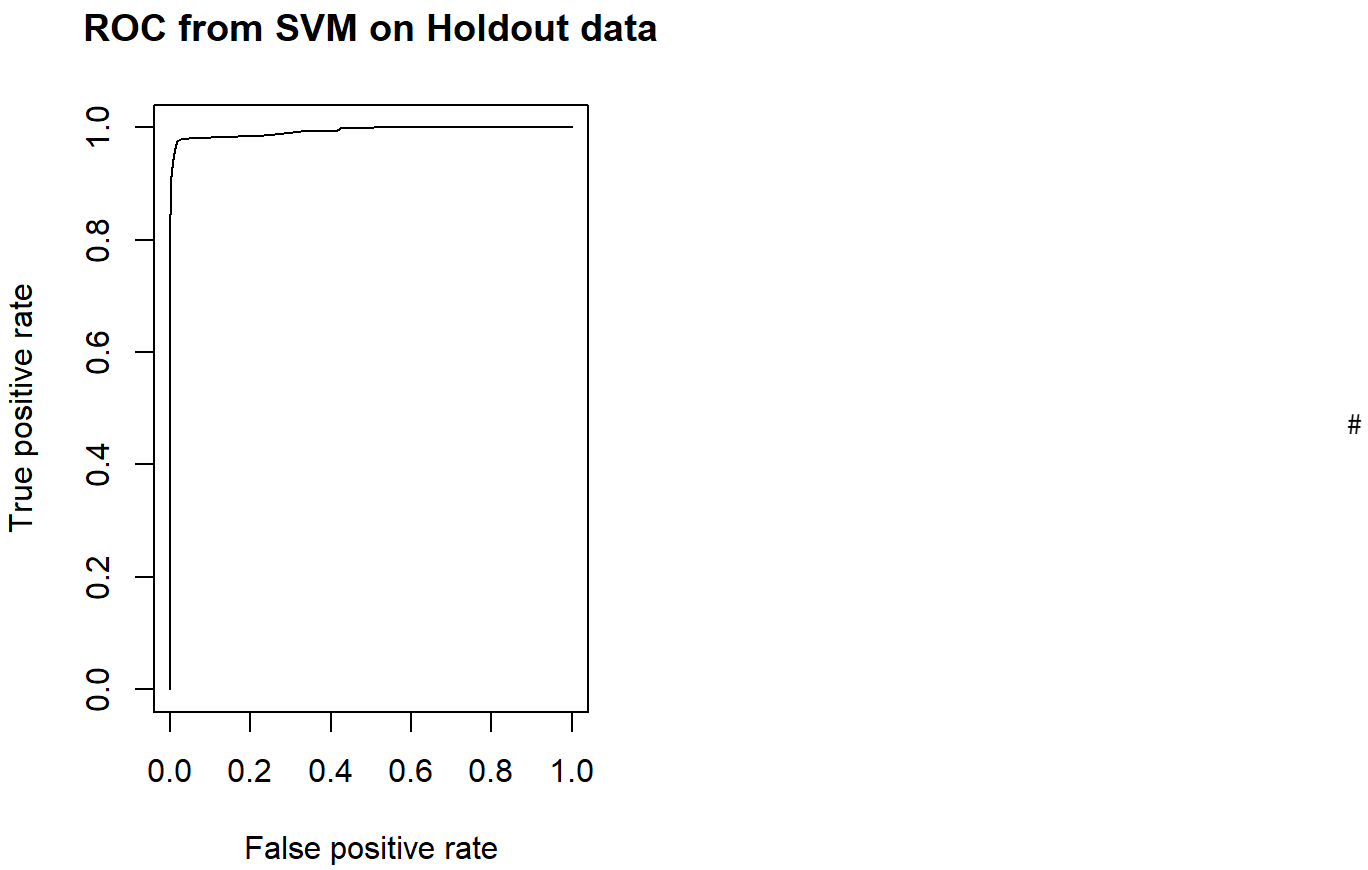
```
library(e1071)


svmfit=svm(Class.Dummy~Red+Green+Blue+Blue^2, data=haiti,kernel="radial",gamma=2,cost=1, s
        cale=TRUE)
#https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-r/
summary(svmfit)
```

```
##
## Call:
## svm(formula = Class.Dummy ~ Red + Green + Blue + Blue^2, data = haiti,
##     kernel = "radial", gamma = 2, cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  4599
```

```
#Encoding the target feature as factor
#haiti$Class.Dummy <- factor(haiti$Class.Dummy, levels = c(0,1))

fitted=attributes(predict(svmfit,holdout.all,decision.values = TRUE)) $decision.values #ta
        kes 30 mins
pred <- predict(svmfit,holdout.all) #takes 30 mins

#table(true=holdout.all[,"Class.Dummy"],pred=predict(tune.out$best.model,newdata=haiti[hai
        ti.test,]))
# svm.table.ho = table(true=holdout.all$Class.Dummy,pred=fitted)

#svm.table.ho <- table(holdout.all$Class.Dummy, pred)#(actual, predicted)
#svm.table.ho

dat.te=data.frame(x=holdout.all[,2:4], y=as.factor(holdout.all$Class.Dummy ))
pred.te=predict(svmfit , newdata =holdout.all)

#need to classify raw probabilities
svm.cutoff = .001
svm.class <- ifelse(pred.te > svm.cutoff, 1, 0)
svm.table.ho <- table(dat.te$y,svm.class) #(actual, predicted)
svm.table.ho
```

```
##    svm.class
##           0       1
##   0  764281 1225416
##   1       0   14480
```

```
#ROC Curves
library (ROCR)
rocplot=function(pred,truth, ...) {
  predob = prediction(pred,truth)
  perf = performance(predob, "tpr" ,"fpr") #ignore the axes labels!! had to switch fpr and
         tpr to make plot correct
  plot(perf,...)}


par(mfrow=c(1,2)) #create one row two columns of figures to plot

rocplot(fitted, holdout.all[,"Class.Dummy"],main="ROC from SVM on Holdout data")
predob = prediction(pred.te,dat.te$y)
perf_val <- performance(predob,"auc")
svm.auc.ho <-perf_val@y.values[[1]]
```



#

Peformance Table for HOLDOUT

```
pt.ho <- setNames(data.frame(matrix(ncol = 8, nrow = 7)), c("Model", "Tuning", "AUROC" ,
         "Threshold" , "Accuracy", "TPR", "FPR" , "Precision"))

pt.ho[1,1] = "Logistic"
pt.ho[2,1] = "LDA"
pt.ho[3,1] = "QDA"
pt.ho[4,1] = "KNN"
pt.ho[5,1] = "PLR"
pt.ho[6,1] = "R.Forest"
pt.ho[7,1] = "SVM"


pt.ho[1,2] = "   *   "  #tuning
pt.ho[1,3] = formatC(lm.auc.ho)           #AUC
pt.ho[1,4] = formatC(lm.cutoff.ho)        #threshold
pt.ho[1,5] = formatC((lm.table.ho[1,1]+lm.table.ho[2,2]) / sum(lm.table.ho)) #Accuracy
pt.ho[1,6] = formatC((lm.table.ho[2,2]) / sum(lm.table.ho[2,]))           #TPR
pt.ho[1,7] = formatC(lm.table.ho[1,2] / sum(lm.table.ho[1,]))             #FPR
pt.ho[1,8] = formatC((lm.table.ho[2,2])/ sum(lm.table.ho[,2]))            #Precision


pt.ho[2,2] = "   *   "  #tuning
pt.ho[2,3] = formatC(lda.auc.ho)           #AUC
pt.ho[2,4] = formatC(lda.cutoff)        #threshold
pt.ho[2,5] = formatC((lda.table.ho[1,1]+lda.table.ho[2,2]) / sum(lda.table.ho)) #Accuracy
pt.ho[2,6] = formatC((lda.table.ho[2,2]) / sum(lda.table.ho[2,]))           #TPR
pt.ho[2,7] = formatC(lda.table.ho[1,2] / sum(lda.table.ho[1,]))             #FPR
pt.ho[2,8] = formatC((lda.table.ho[2,2])/ sum(lda.table.ho[,2]))            #Precision



pt.ho[3,2] = "   *   "  #tuning
pt.ho[3,3] = formatC(qda.auc.ho)           #AUC
pt.ho[3,4] = formatC(qda.cutoff.ho)        #threshold
pt.ho[3,5] = formatC((qda.table.ho[1,1]+qda.table.ho[2,2]) / sum(qda.table.ho)) #Accuracy
pt.ho[3,6] = formatC((qda.table.ho[2,2]) / sum(qda.table.ho[2,]))           #TPR
pt.ho[3,7] = formatC(qda.table.ho[1,2] / sum(qda.table.ho[1,]))             #FPR
pt.ho[3,8] = formatC((qda.table.ho[2,2])/ sum(qda.table.ho[,2]))            #Precision




pt.ho[4,2] = "k= 5    "  #tuning
pt.ho[4,3] = formatC(knn.auc.ho)           #AUC
pt.ho[4,4] = formatC(.5)        #threshold
pt.ho[4,5] = formatC((knn.table.ho[1,1]+knn.table.ho[2,2]) / sum(knn.table.ho)) #Accuracy
pt.ho[4,6] = formatC((knn.table.ho[2,2]) / sum(knn.table.ho[2,]))           #TPR
pt.ho[4,7] = formatC(knn.table.ho[1,2] / sum(knn.table.ho[1,]))             #FPR
pt.ho[4,8] = formatC((knn.table.ho[2,2])/ sum(knn.table.ho[,2]))            #Precision


pt.ho[5,2] = "lambda=8.965e-05 alpha=.05"  #tuning
```

```
pt.ho[5,3] = formatC(plr.auc.ho)           #AUC
pt.ho[5,4] = formatC(.5)        #threshold
pt.ho[5,5] = formatC((plr.table.ho[1,1]+plr.table.ho[2,2]) / sum(plr.table.ho)) #Accuracy
pt.ho[5,6] = formatC((plr.table.ho[2,2]) / sum(plr.table.ho[2,]))           #TPR
pt.ho[5,7] = formatC(plr.table.ho[1,2] / sum(plr.table.ho[1,]))             #FPR
pt.ho[5,8] = formatC((plr.table.ho[2,2])/ sum(plr.table.ho[,2]))            #Precision


pt.ho[6,2] = "mtry = 3 ntree=500"  #tuning
pt.ho[6,3] = formatC(rf.auc.ho)           #AUC
pt.ho[6,4] = "p1 > p0"        #threshold
pt.ho[6,5] = formatC((rf.table.ho[1,1]+rf.table.ho[2,2]) / sum(rf.table.ho)) #Accuracy
pt.ho[6,6] = formatC((rf.table.ho[2,2]) / sum(rf.table.ho[2,]))           #TPR
pt.ho[6,7] = formatC(rf.table.ho[1,2] / sum(rf.table.ho[1,]))             #FPR
pt.ho[6,8] = formatC((rf.table.ho[2,2])/ sum(rf.table.ho[,2]))            #Precision

pt.ho[7,2] = "cost=1, gamma=2, kernel=radial"  #tuning
pt.ho[7,3] = formatC(svm.auc.ho)           #AUC
pt.ho[7,4] = formatC(svm.cutoff)        #threshold
pt.ho[7,5] = formatC((svm.table.ho[1,1]+svm.table.ho[2,2]) / sum(svm.table.ho)) #Accuracy
pt.ho[7,6] = formatC((svm.table.ho[2,2]) / sum(svm.table.ho[2,]))           #TPR
pt.ho[7,7] = formatC(svm.table.ho[1,2] / sum(svm.table.ho[1,]))             #FPR
pt.ho[7,8] = formatC((svm.table.ho[2,2])/ sum(svm.table.ho[,2]))            #Precision

#kable(pt.ho[1:5,], width="20cm" , caption = "Performance Table")
pt.ho
```

```
##      Model                         Tuning  AUROC Threshold Accuracy    TPR
## 1 Logistic                           *    0.9994     0.034   0.8904      1
## 2      LDA                           *    0.9921     0.015   0.9702 0.9457
## 3      QDA                           *    0.9915       0.5    0.996 0.6946
## 4      KNN                        k= 5    0.9413       0.5   0.9764 0.9057
## 5      PLR     lambda=8.965e-05 alpha=.05 0.9921       0.5   0.9119 0.9999
## 6 R.Forest             mtry = 3 ntree=500 0.9921   p1 > p0   0.9923 0.7135
## 7      SVM cost=1, gamma=2, kernel=radial 0.9932     0.001   0.3886      1
##        FPR Precision
## 1   0.1104   0.06183
## 2  0.02963    0.1885
## 3 0.001835    0.7337
## 4  0.02304    0.2224
## 5  0.08878   0.07576
## 6 0.005634    0.4796
## 7   0.6159   0.01168
```

7 Conclusions Part II

7.1 Concusion #1 - A discussion of the best performing algorithm(s) in the cross-validation and hold-out data.

In the cross-validation, I reviewed the individual k-fold errors for each of the models. The model that has the lowest test errors for each of the folds (prior to adding random forest and SVM models) was KNN, and it had the highest accuracy at 99.76% and True Positive Rate (TPR) of 95.21% and a False Postive Rate (FPR) of .0009.

However, the type II error appears to the lowest for the logistic model. The accuracy is very high at 98.2%. There is some concern with the precision being low (62.4%) based on the resources required or wasted for all classifications not being precise. A cost/benefit analysis would help understand this, if data regarding costs of food and supplies were available.

Once Random Forest and SVM were added, the TPR for these models result in 100% moving them to the top of the consideration set.

When reviewing the best applied algorithm to the holdout data, the KNN model had a high accuracy of 97.64% with a TPR of ~91%. This is concerning as I would prefer applied results to have a high TPR, such as the applied PLR method.

7.2 Conclusion #2 - A discussion or analysis justifying why your findings above are compatible or reconcilable.

In the test case, KNN was the original “winner” and continued to perform well when applied to the holdout data. All of the models testing within this project had high cross-validation accuracy, except for SVM. My main criteria for choosing the best model was based on the ability to predict with 100% TPR all of the blue tarps. This narrowed the selection between PLR, Random Forest, and SVM based on the cross-validation data. Random Forest TPR decreased when applied to the holdout data. While KNN and PLR remained high at .90 and .99, respectively. However, SVM became the best model with 100% TPR on the holdout data! Therefore, the final three models for consideration were PLR, Random Forest and SVM. Random forest had the highest accuracy in the test scenario and in the holdout application, demonstrating compatibility and stability. The k=5 value determined for KNN helps avoid overfitting but even though it won on the cross-validation, the accuracy went to 94%, and TPR 90%. The PLR is a simplified model with fast computational time, but it’s precision was low in both CV and holdout data.

7.3 Conclusion #3 - A recommendation and rationale regarding which algorithm to use for detection of blue tarps.

The algorithm I recommend using for detection of blue tarps is a Support Vector Machine (SVM) with the following features: Red, Green and Blue pixels, with the inclusion of a polynomial function to the 2nd degree on “Blue”. The rationale for choosing the SVM method is because it handles well with a smaller number of features. It was able to develop a clear margin of separation between blue tarps and non-blue tarps, even though the blue tarps were less frequent than others in the datasets. In our data, in both the original “Haiti” data, and the “Holdout Data”, the representation of Blue Tarps was less frequent than no blue tarp at 3.2% in the original Haiti data and less than one percent (< 1%) in the holdout data. SVM works well with non-linear data, and has a lower risk of overfitting. The SVM had a perfect TPR in cross-validation and in the holdout dataset. The accuracy was low due to a higher FPR, and low precision, but as described in conclusion 4, TPR was of most concern. The trade off between the amount of time to process an SVM models makes up for it’s ability to accurately predict blue tarps in a disaster recovery situation such as the Haiti earthquakes.

7.4 Conclusion #4 - A discussion of the relevance of the metrics calculated in the tables to this application context.

This data is about saving lives. In reviewing the error results for each of the model methods, and given the circumstances of the application of this data, I feel it is much better to predict there is a tarp when there wasn't (100% True, some False), than wrongly predict that there was not a tarp when there actually was. So in selecting best model and cut-off values, I was relaxed on the False positive, Type I error. However, I want to reduce the Type II error, where I would predict there is not a tarp, when in fact there is. The most important metric to me was the TPR.

Overall accuracy is a good measure, but is not my first choice when evaluation these methods due to my decision to relax the Type I error concern. It is better to predict a tarp, when one doesn't exist, than not predict one, where there is one. Accuracy does not help us determine this.

Precision is misleading in this model evaluation because the lack of precision in some models was due to the large number of false positives my models produced. The PLR model and SVM model had the highest FPR's compared to the other models which impact the precision. If only looking at precision, one may not choose the SVM model, missing the benefits that SVM provides.

7.5 Conclusion #5

I recommend spending time examining the specific observations in which Type II error did occur (for some test results k-folds there were 4 of them, other more). Since the thresholds were already carefully chosen, I would recommend examining what the RGB values for those cases that were incorrectly classified to look for additional patterns. Are they always the same RGB values? Are they a different relationship than the overall average relationship of RGB to BlueTarp prediction?

7.6 Conclusion #6

To improve model predictions, I believe that having additional data such as the latitude and longitude of where each of the tarps are may help in understanding the risk of the Type II error. For example, is it wasting resources to go to an area without tarps? How does the distance from other predicted tarps impact the next observation? How could applying clustering methods (found in Chapter 10 of the textbook) be used to group observations of predicted tarps based on latitude and longitude? Other data, such as previous population density may also help "weight" the observations towards "more likely to have tarp" versus not and could be helpful in the model.

8 Sources

8.1 General overall R coding:

https://www.rdocumentation.org/ (https://www.rdocumentation.org/) (for basic understanding of function parameters and inputs) https://stats.stackexchange.com/questions/4040/r-compute-correlation-by-group (https://stats.stackexchange.com/questions/4040/r-compute-correlation-by-group) –> https://www.sfu.ca/~mjbrydon/tutorials/BAinR/recode.html (https://www.sfu.ca/~mjbrydon/tutorials/BAinR/recode.html) –> http://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/#k-fold-cross-validation (http://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/#k-fold-cross-validation) https://stackoverflow.com/questions/32712301/create-empty-data-frame-with-column-names-by-assigning-a-string-vector/32712555 (https://stackoverflow.com/questions/32712301/create-empty-data-frame-with-column-names-by-assigning-a-string-vector/32712555)

8.2 Specific Model Help:

Throughout the r chunks, I have noted next to lines of code where I referenced the following specific sources. However, I have also summarized them here:

James, Witten, Hastie, and Tibshirani. An Introduction to Statistical Learning with Applications in R. Springer, 2013.
https://rpubs.com/Kushan/295412 (https://rpubs.com/Kushan/295412) https://rstudio-pubs-static.s3.amazonaws.com/16444_caf85a306d564eb490eebdbaf0072df2.html (https://rstudio-pubs-static.s3.amazonaws.com/16444_caf85a306d564eb490eebdbaf0072df2.html)
http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/#r-functions (http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/#r-functions)
https://daviddalpiaz.github.io/r4sl/elastic-net.html (https://daviddalpiaz.github.io/r4sl/elastic-net.html)
https://stackoverflow.com/questions/46124424/how-can-i-draw-a-roc-curve-for-a-randomforest-model-with-three-classes-in-r (https://stackoverflow.com/questions/46124424/how-can-i-draw-a-roc-curve-for-a-randomforest-model-with-three-classes-in-r) https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-r/ (https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-r/)