Project: Part 2

Marin Lolic

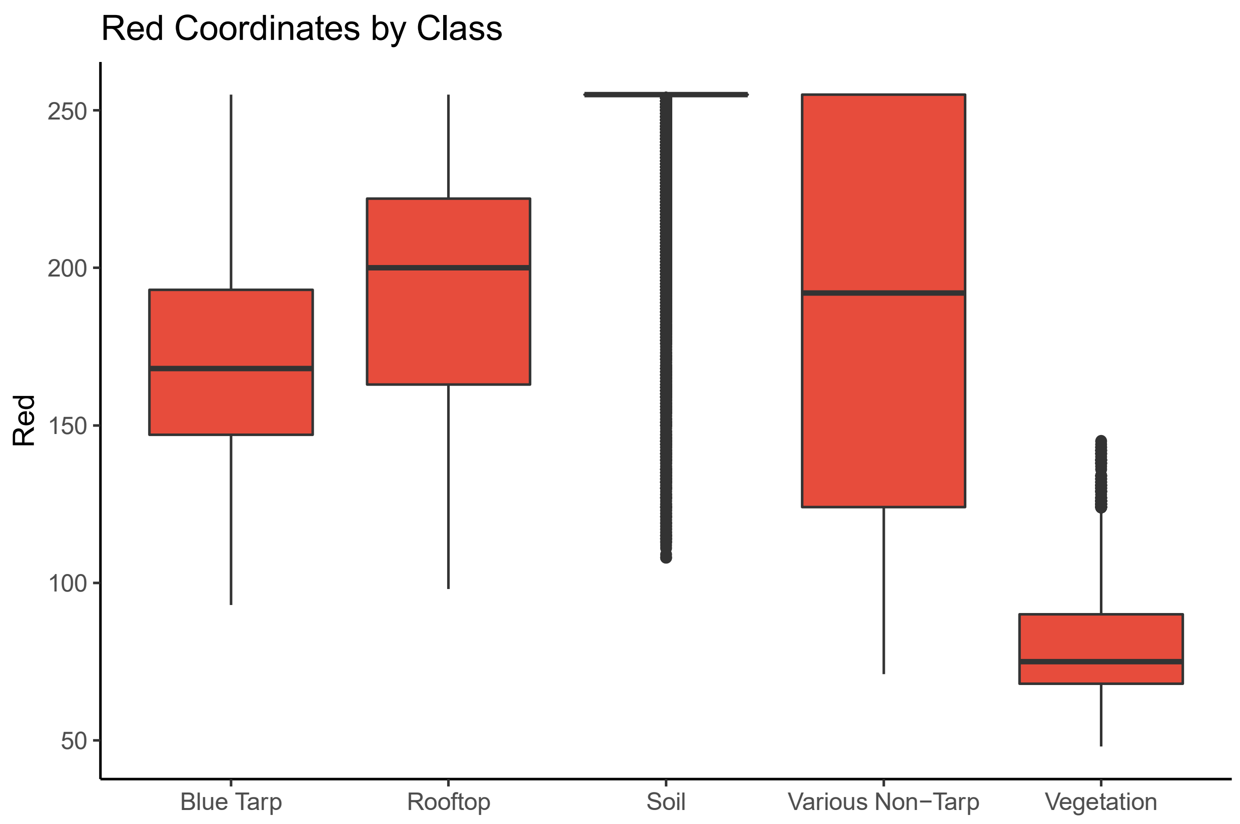
4/17/2022

Initial Set-up

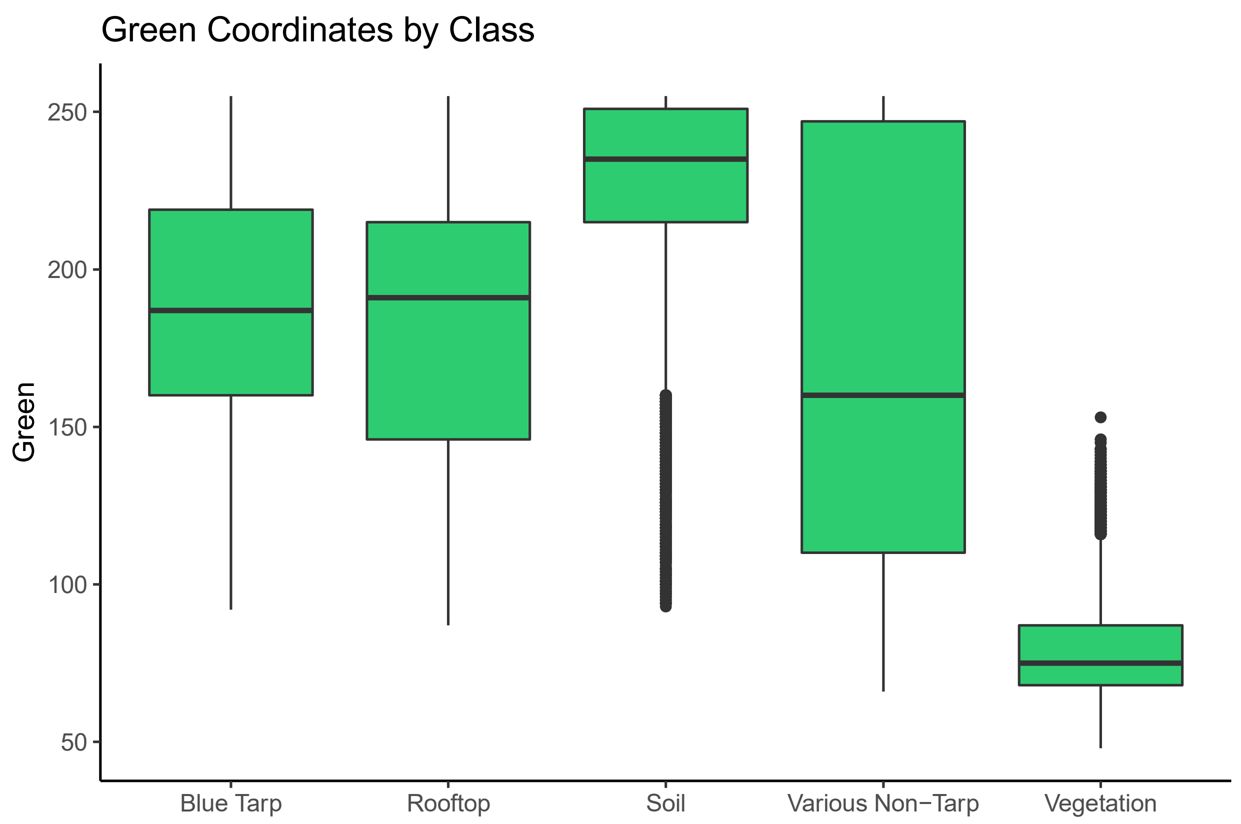
```
library(caret)
library(tidyverse)
library(knitr)
library(kernlab)
library(doParallel)

cl <- makePSOCKcluster(8)
registerDoParallel(cl)

Haiti <- read_csv("HaitiPixels.csv")
Haiti2 <- mutate(Haiti, BlueTarp = ifelse(Class == "Blue Tarp", 1, 0))
Haiti2$BlueTarp <- factor(Haiti2$BlueTarp)
Real <- Haiti2$BlueTarp
```

Exploratory Data Analysis

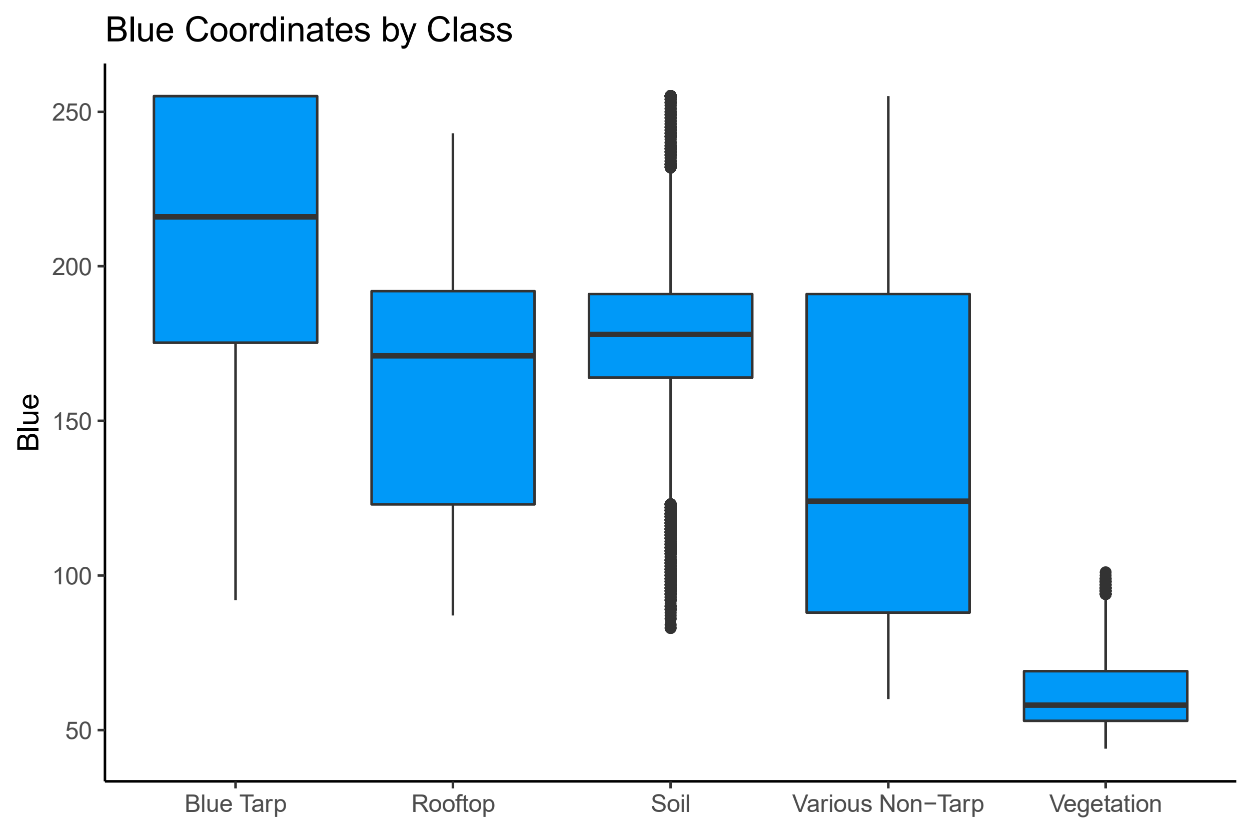
```
ggplot(Haiti, aes(x = Class, y = Red)) +
    geom_boxplot(fill = "#e74c3c") +
    theme_classic() +
    ggtitle("Red Coordinates by Class") +
    xlab("")
```



```
ggplot(Haiti, aes(x = Class, y = Green)) +
    geom_boxplot(fill = "#2ecc71") +
    theme_classic() +
    ggtitle("Green Coordinates by Class") +
    xlab("")
```



```r
ggplot(Haiti, aes(x = Class, y = Blue)) +
    geom_boxplot(fill = "#0099f8") +
    theme_classic() +
    ggtitle("Blue Coordinates by Class") +
    xlab("")
```



Not surprisingly, the blue tarps scored very high on the blue coordinates. However, three of the other classes had at least some data points that also scored highly. In general, there was overlap in coordinate colors for most of the classes, suggesting that no classifier will work perfectly here.

Baseline Accuracy (No-Information Rate)

```
Baseline <- sum(Haiti2$BlueTarp == "0")/nrow(Haiti2)
Baseline
```

```
## [1] 0.9680271
```

Even with no model, we can achieve 96.8% accuracy by predicting that no pixel is a blue tarp. However, this would not be very useful in practice.

K-Nearest Neighbors

```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 allowParallel = TRUE)

KNNModel <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               tuneGrid=data.frame(k=c(1,3,5,7,9,11,13,15,17,19,21,23,25)),
               method = 'knn',
               trControl = trControl)
```

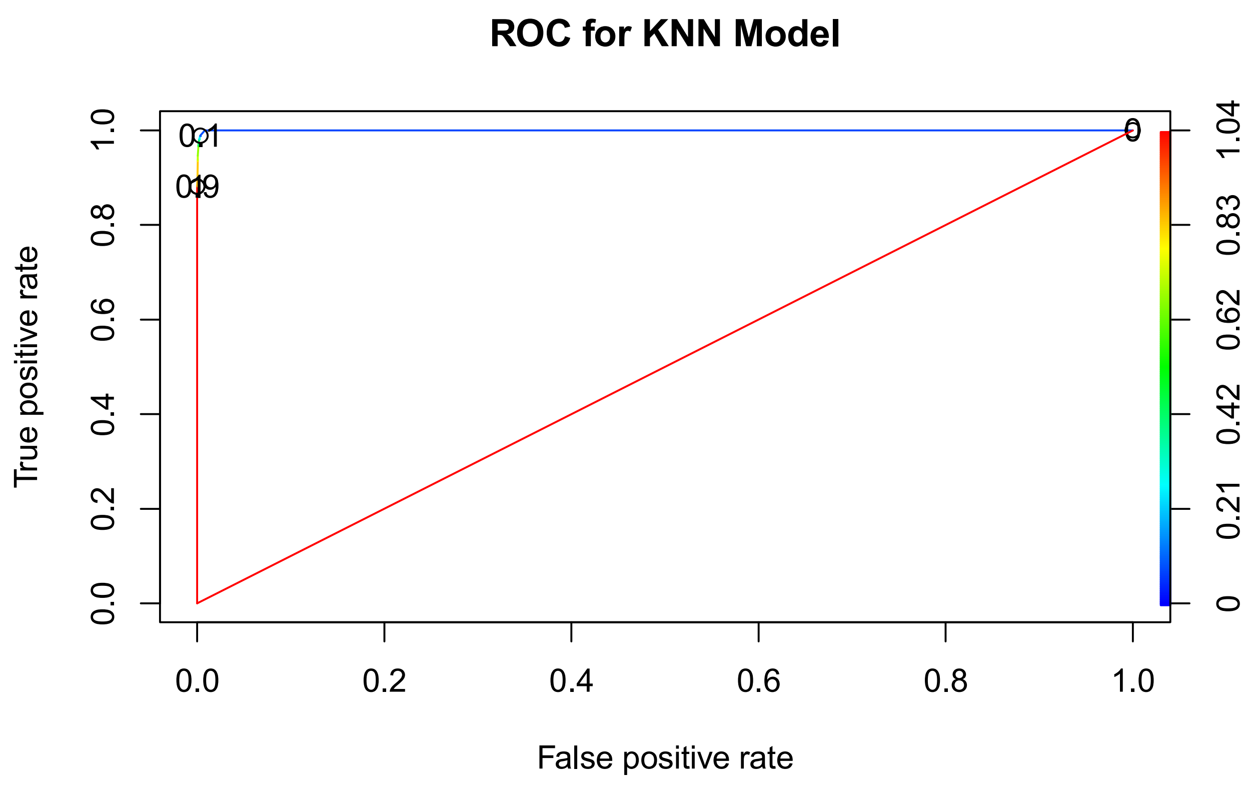
```r
set.seed(1)
KNNPred <- predict(KNNModel, newdata = Haiti2)
KNNOutput <- caret::confusionMatrix(data = KNNPred, reference = Real)
KNNOutput
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61150    65
##          1    69  1957
##
##                Accuracy : 0.9979
##                  95% CI : (0.9975, 0.9982)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9658
##
##  Mcnemar's Test P-Value : 0.7955
```

```
##
##             Sensitivity : 0.9989
##             Specificity : 0.9679
##          Pos Pred Value : 0.9989
##          Neg Pred Value : 0.9659
##              Prevalence : 0.9680
##          Detection Rate : 0.9669
##    Detection Prevalence : 0.9680
##       Balanced Accuracy : 0.9834
##
##        'Positive' Class : 0
##
```

```
KNNPerf <- c("K-Nearest Neighbors", "K = 5", 0.9999, "", 0.9979, 0.9989, 1 - 0.9679,
             0.9989)
```

```
Real3 <- as.numeric(Real)
KNNPred2 <- predict(KNNModel, newdata = Haiti2, type = 'prob')
KNNdob <- ROCR::prediction(KNNPred2['1'], Real3)
KNNroc <- ROCR::performance(KNNdob, measure='tpr', x.measure='fpr')

plot(KNNroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for KNN Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```
KNNauc <- ROCR::performance(KNNdob, measure = "auc")
print(KNNauc@y.values)
```

```
## [[1]]
## [1] 0.9998582
```

Linear Discriminant Analysis

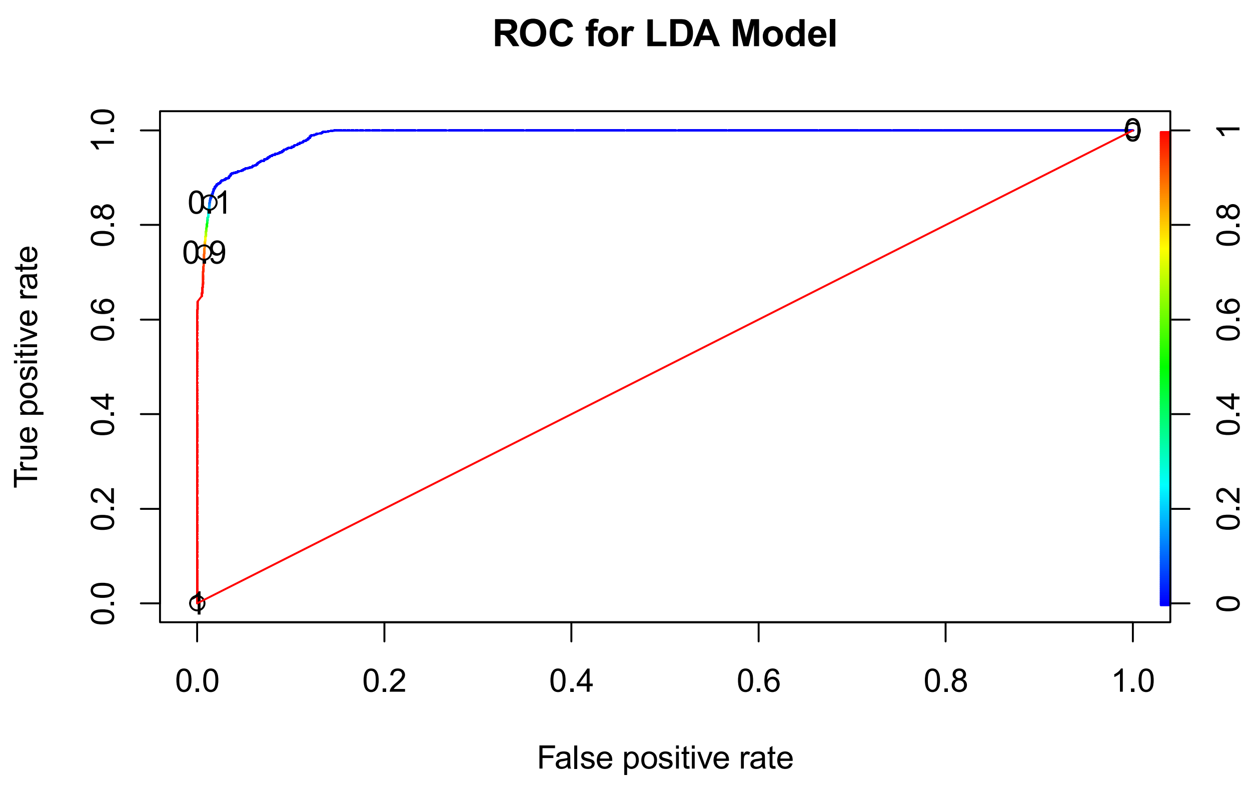
```
trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 allowParallel = TRUE)

LDAModel <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'lda',
               trControl = trControl)
```

```
LDAPred <- predict(LDAModel, newdata = Haiti2)
LDAOutput <- caret::confusionMatrix(data = LDAPred, reference = Real)
LDAOutput
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 60607   402
##          1   612  1620
##
##                Accuracy : 0.984
##                  95% CI : (0.983, 0.9849)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7534
##
##  Mcnemar's Test P-Value : 5.26e-11
##
##             Sensitivity : 0.9900
##             Specificity : 0.8012
##          Pos Pred Value : 0.9934
##          Neg Pred Value : 0.7258
##              Prevalence : 0.9680
##          Detection Rate : 0.9583
##    Detection Prevalence : 0.9647
##       Balanced Accuracy : 0.8956
##
##        'Positive' Class : 0
##
```

```
LDAPerf <- c("Linear Discriminant Analysis", "", 0.9889, 0.50, 0.9840, 0.9900,
             1 - 0.8012, 0.9934)
```

```
LDAPred2 <- predict(LDAModel, newdata = Haiti2, type = 'prob')
LDAdob <- ROCR::prediction(LDAPred2['1'], Real3)
LDAroc <- ROCR::performance(LDAdob, measure='tpr', x.measure='fpr')

plot(LDAroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for LDA Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```
LDAauc <- ROCR::performance(LDAdob, measure = "auc")
print(LDAauc@y.values)
```

```
## [[1]]
## [1] 0.9888768
```

Quadratic Discriminant Analysis

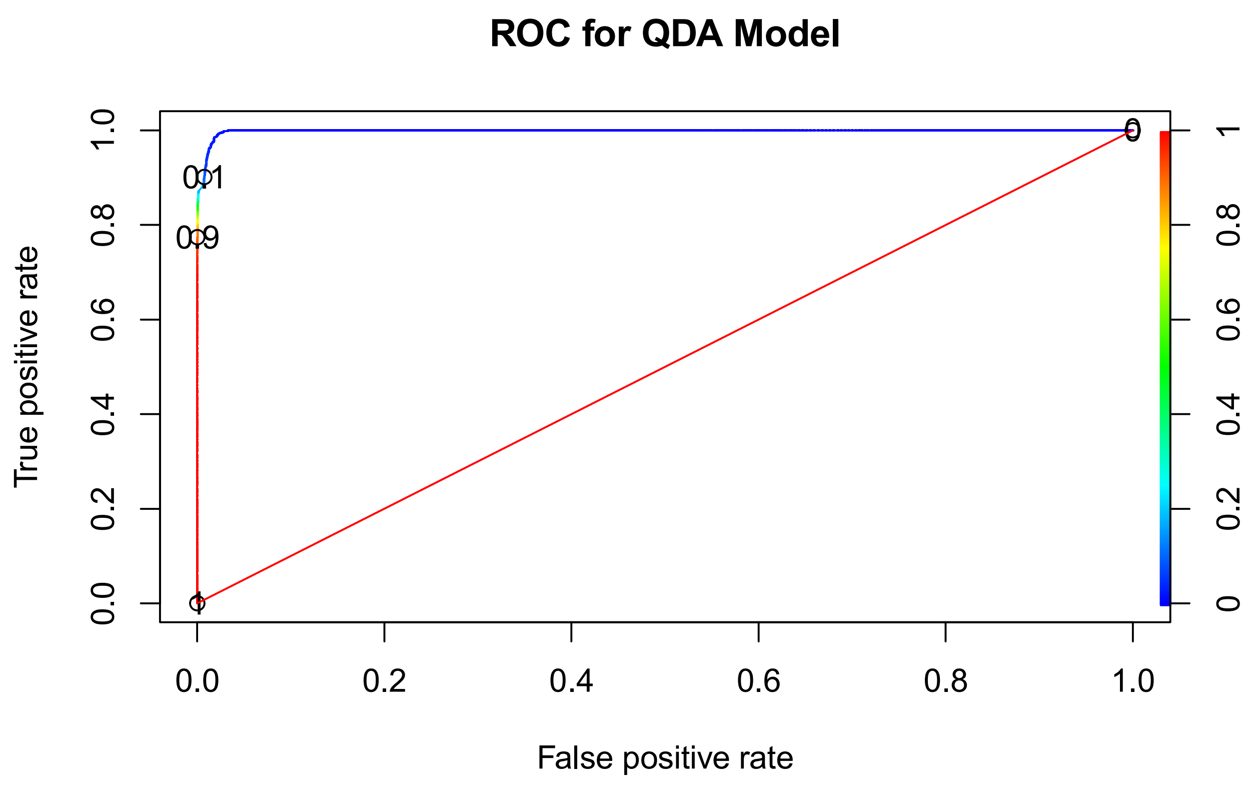
```r
trControl <- caret::trainControl(method = 'cv',
                                number = 10,
                                allowParallel = TRUE)

QDAModel <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'qda',
               trControl = trControl)
```

```r
QDAPred <- predict(QDAModel, newdata = Haiti2)
QDAOutput <- caret::confusionMatrix(data = QDAPred, reference = Real)
QDAOutput
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61201   323
##          1    18  1699
##
##                Accuracy : 0.9946
##                  95% CI : (0.994, 0.9952)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.906
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9997
##             Specificity : 0.8403
##          Pos Pred Value : 0.9948
##          Neg Pred Value : 0.9895
##              Prevalence : 0.9680
##          Detection Rate : 0.9677
##    Detection Prevalence : 0.9728
##       Balanced Accuracy : 0.9200
##
##        'Positive' Class : 0
##
```

```
QDAPerf <- c("Quadratic Discriminant Analysis", "", 0.9982, 0.50, 0.9946,
             0.9997, 1 - 0.8403, 0.9948)
```

```
QDAPred2 <- predict(QDAModel, newdata = Haiti2, type = 'prob')
QDAdob <- ROCR::prediction(QDAPred2['1'], Real3)
QDAroc <- ROCR::performance(QDAdob, measure='tpr', x.measure='fpr')

plot(QDAroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for QDA Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```
QDAauc <- ROCR::performance(QDAdob, measure = "auc")
print(QDAauc@y.values)
```

```
## [[1]]
## [1] 0.9982175
```

Logistic Regression

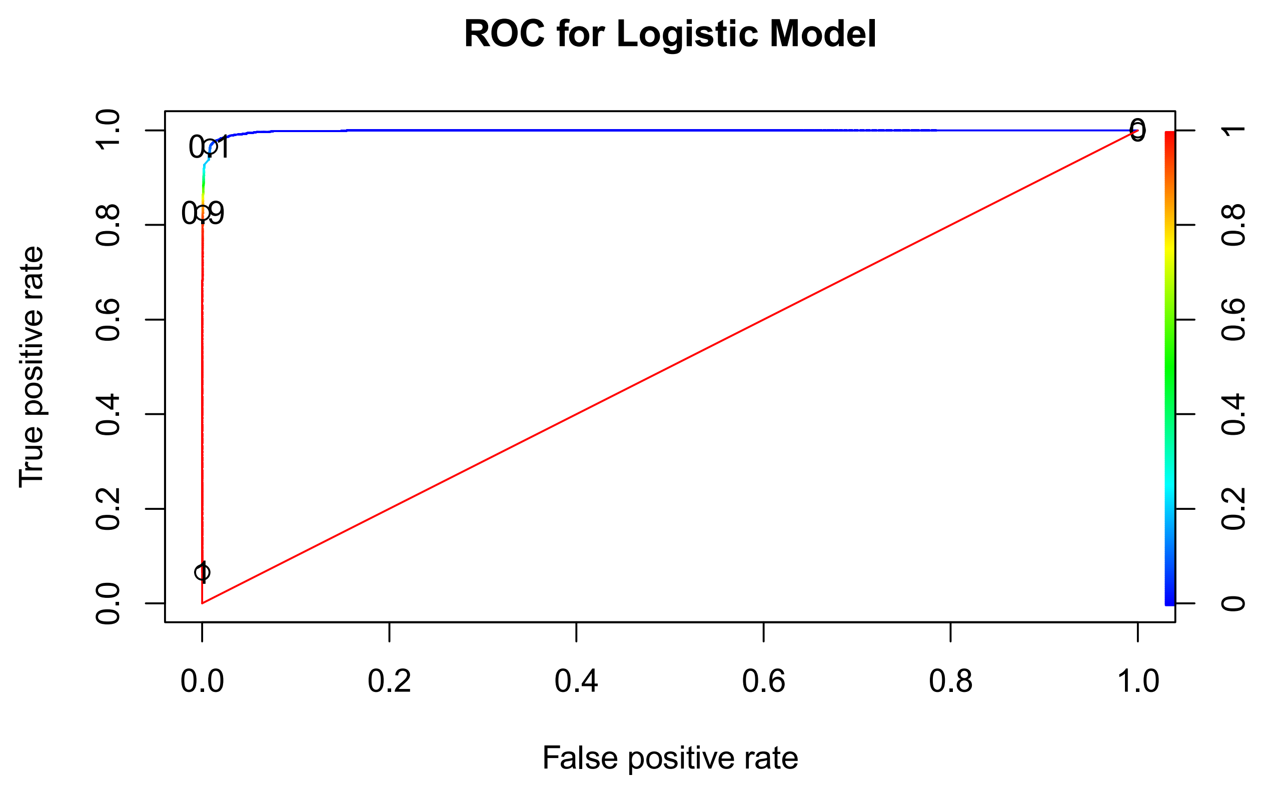
```r
trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 allowParallel = TRUE)

LogModel <- caret::train(BlueTarp ~ Red + Green + Blue,
                data = Haiti2,
                method = 'glm',
                family = 'binomial',
                trControl = trControl)
```

```r
LogPred <- predict(LogModel, newdata = Haiti2)
LogOutput <- caret::confusionMatrix(data = LogPred, reference = Real)
LogOutput
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61153   232
##          1    66  1790
##
##                Accuracy : 0.9953
##                  95% CI : (0.9947, 0.9958)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9207
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9989
##             Specificity : 0.8853
##          Pos Pred Value : 0.9962
##          Neg Pred Value : 0.9644
##              Prevalence : 0.9680
##          Detection Rate : 0.9670
##    Detection Prevalence : 0.9707
##       Balanced Accuracy : 0.9421
##
##        'Positive' Class : 0
##
```

```
LogPerf <- c("Logistic Regression", "", 0.9985, 0.50, 0.9953, 0.9989,
             1 - 0.8853, 0.9934)
```

```
LogPred2 <- predict(LogModel, newdata = Haiti2, type = 'prob')
Logdob <- ROCR::prediction(LogPred2['1'], Real3)
Logroc <- ROCR::performance(Logdob, measure='tpr', x.measure='fpr')

plot(Logroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for Logistic Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```r
Logauc <- ROCR::performance(Logdob, measure = "auc")
print(Logauc@y.values)
```

```
## [[1]]
## [1] 0.9985069
```

Penalized Logistic Regression

```
Haiti3 <- Haiti2
levels(Haiti3$BlueTarp) <- c('unrelated', 'linked')

lambdas <- 10^seq(0, -4, by = -0.1)

trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 savePredictions = TRUE,
                                 classProbs = TRUE,
                                 allowParallel = TRUE)

tuneGrid <- expand.grid(lambda=lambdas, alpha = 0.5)

PLogModel <- caret::train(BlueTarp ~ Red + Green + Blue,
                 data = Haiti3,
                 method = 'glmnet',
                 family = 'binomial',
                 thresh = .5,
                 trControl = trControl,
                 tuneGrid = tuneGrid)
```

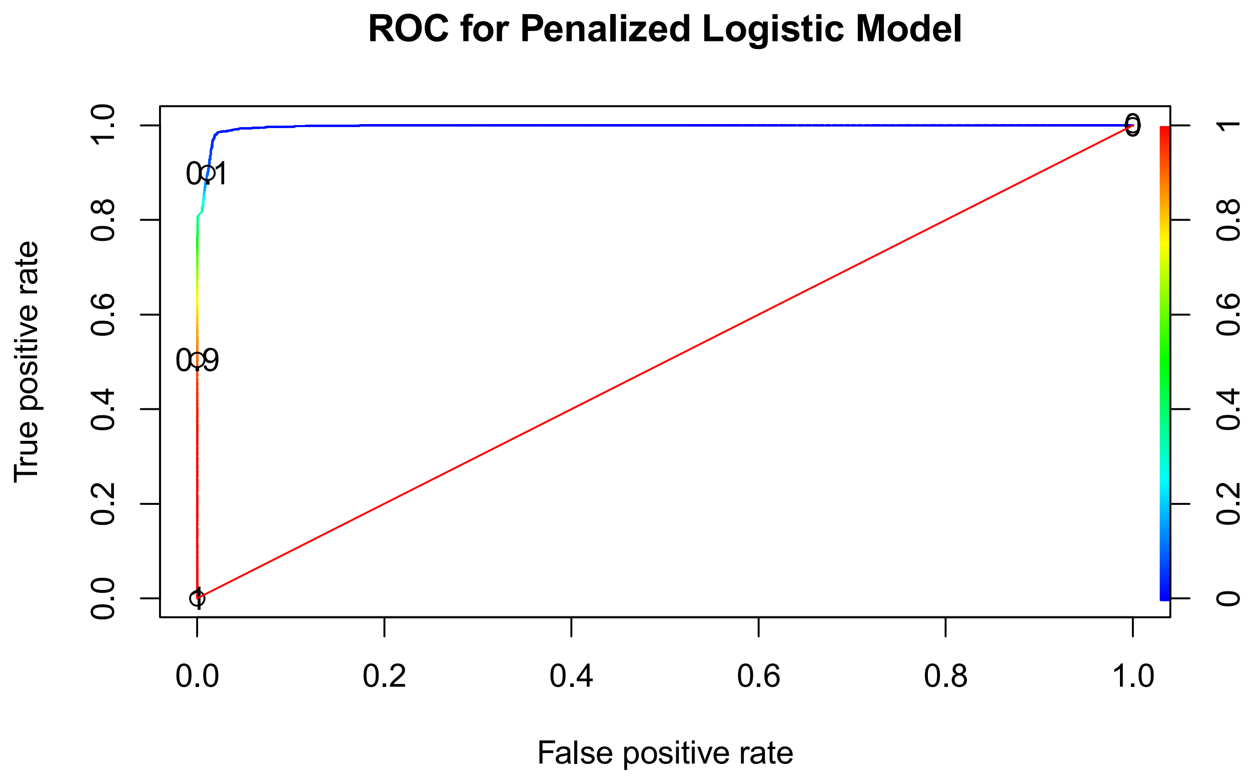
```
Real2 <- Real
levels(Real2) <- c('unrelated', 'linked')

PLogPred <- predict(PLogModel, newdata = Haiti3)
PLogOutput <- caret::confusionMatrix(data = PLogPred, reference = Real2)
PLogOutput
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  unrelated linked
##   unrelated     61218    499
##   linked            1   1523
##
```

```
##                Accuracy : 0.9921
##                  95% CI : (0.9914, 0.9928)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.855
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.7532
##          Pos Pred Value : 0.9919
##          Neg Pred Value : 0.9993
##              Prevalence : 0.9680
##          Detection Rate : 0.9680
##    Detection Prevalence : 0.9759
##       Balanced Accuracy : 0.8766
##
##        'Positive' Class : unrelated
##
```

```
PLogPerf <- c("Penalized Logistic Regression", "Lambda = 0.001", 0.9971, 0.50, 0.9921,
              1.0000, 1 - 0.7532 , 0.9919)
```

```
PLogPred2 <- predict(PLogModel, newdata = Haiti2, type = 'prob')
PLogdob <- ROCR::prediction(PLogPred2[2], Real3)
PLogroc <- ROCR::performance(PLogdob, measure='tpr', x.measure='fpr')

plot(PLogroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for Penalized Logistic Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```r
PLogauc <- ROCR::performance(PLogdob, measure = "auc")
print(PLogauc@y.values)
```

```
## [[1]]
## [1] 0.9971498
```

Random Forest

```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                number = 10,
                                allowParallel = TRUE)

rfgrid <- expand.grid(mtry = c(1, 2, 3))

RFModel1 <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'rf',
               ntree = 100,
               tuneGrid = rfgrid,
               trControl = trControl)
```

```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                number = 10,
                                allowParallel = TRUE)

rfgrid <- expand.grid(mtry = c(1, 2, 3))

RFModel2 <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'rf',
               ntree = 250,
               tuneGrid = rfgrid,
               trControl = trControl)
```

```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                number = 10,
                                allowParallel = TRUE)
```

```r
rfgrid <- expand.grid(mtry = c(1, 2, 3))

RFModel3 <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'rf',
               ntree = 500,
               tuneGrid = rfgrid,
               trControl = trControl)
```

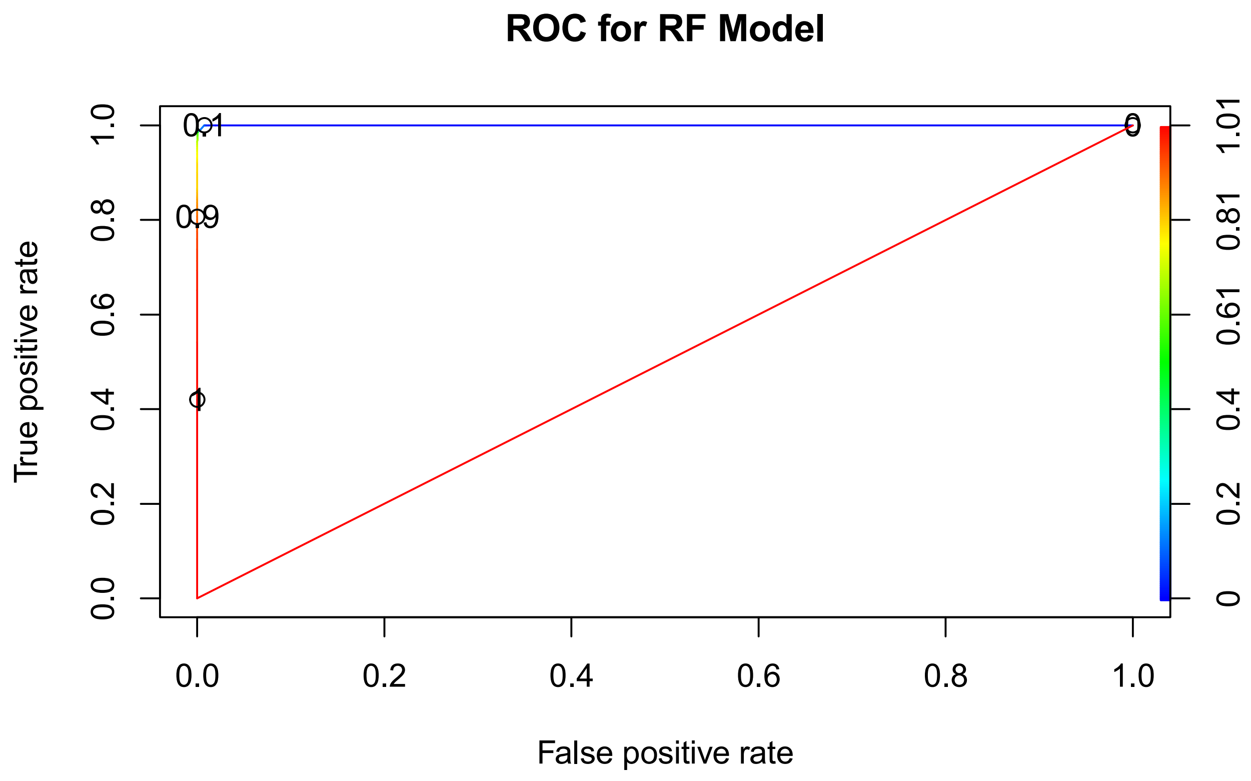
```r
set.seed(1)
RFPred <- predict(RFModel1, newdata = Haiti2)
RFOutput <- caret::confusionMatrix(data = RFPred, reference = Real)
RFOutput
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61211    29
##          1     8  1993
##
##                Accuracy : 0.9994
##                  95% CI : (0.9992, 0.9996)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##
##  Mcnemar's Test P-Value : 0.001009
##
##             Sensitivity : 0.9999
##             Specificity : 0.9857
##          Pos Pred Value : 0.9995
##          Neg Pred Value : 0.9960
##              Prevalence : 0.9680
##          Detection Rate : 0.9679
##    Detection Prevalence : 0.9684
##       Balanced Accuracy : 0.9928
```

```
##
##        'Positive' Class : 0
##
```

```
RFPerf <- c("Random Forest", "Mtry = 1, Trees = 100", 0.9999, 0.50, 0.9994, 0.9999,
             1 - 0.9857, 0.9995)
```

```
RFPred2 <- predict(RFModel1, newdata = Haiti2, type = 'prob')
RFdob <- ROCR::prediction(RFPred2['1'], Real3)
RFroc <- ROCR::performance(RFdob, measure='tpr', x.measure='fpr')

plot(RFroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for RF Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```r
RFauc <- ROCR::performance(RFdob, measure = "auc")
print(RFauc@y.values)
```

```
## [[1]]
## [1] 0.9999443
```

Support Vector Machines

```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 allowParallel = TRUE)

SVMModelL <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'svmLinear',
               tuneGrid = expand.grid(C = c(0.01, 0.1, 1, 10, 100)),
               trControl = trControl)
```

```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 allowParallel = TRUE,
                                 classProbs = TRUE)

SVMModelR <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti3,
               method = 'svmRadial',
               tuneGrid = expand.grid(C = c(0.01, 0.1, 1, 10, 100),
                                      sigma = c(0.5, 1, 2, 3, 4)),
               trControl = trControl)
```

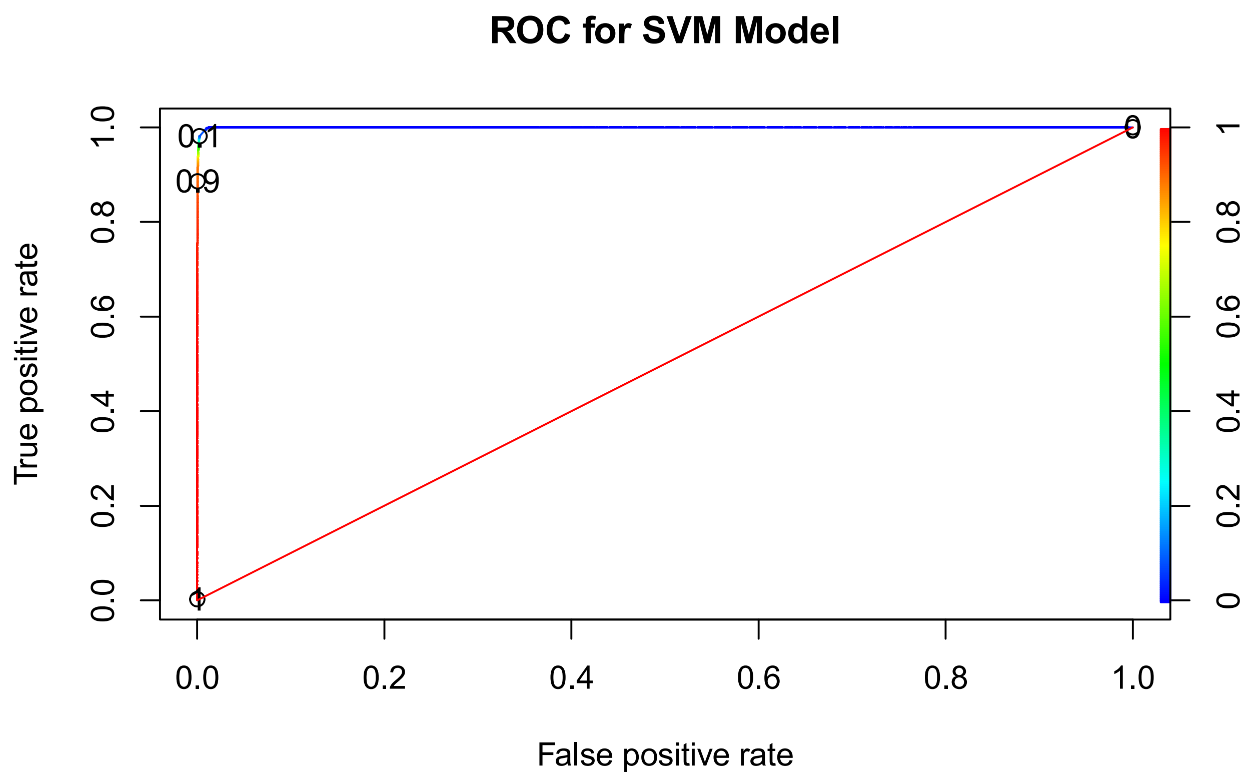
```r
set.seed(1)

trControl <- caret::trainControl(method = 'cv',
                                 number = 10,
                                 allowParallel = TRUE)

SVMModelP <- caret::train(BlueTarp ~ Red + Green + Blue,
               data = Haiti2,
               method = 'svmPoly',
               tuneGrid = expand.grid(C = c(0.01, 0.1, 1, 10, 100),
                                      scale = c(0.5, 1, 2, 3, 4),
                                      degree = c(2, 3, 4)),
               trControl = trControl)
```

```
set.seed(1)
SVMPred <- predict(SVMModelR, newdata = Haiti2)
SVMOutput <- caret::confusionMatrix(data = SVMPred, reference = Real2)
SVMOutput
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  unrelated linked
##   unrelated     61146     84
##   linked           73   1938
##
##                Accuracy : 0.9975
##                  95% CI : (0.9971, 0.9979)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9598
##
##  Mcnemar's Test P-Value : 0.4248
##
##             Sensitivity : 0.9988
##             Specificity : 0.9585
##          Pos Pred Value : 0.9986
##          Neg Pred Value : 0.9637
##              Prevalence : 0.9680
##          Detection Rate : 0.9669
##    Detection Prevalence : 0.9682
##       Balanced Accuracy : 0.9786
##
##        'Positive' Class : unrelated
##
```

```
SVMPerf <- c("Support Vector Machine", "Radial, C = 100, Sigma = 3", 0.9998,
             0.50, 0.9975, 0.9988,1 - 0.9585, 0.9986)
```

```
SVMPred2 <- predict(SVMModelR, newdata = Haiti2, type = 'prob')
SVMdob <- ROCR::prediction(SVMPred2['linked'], Real3)
SVMroc <- ROCR::performance(SVMdob, measure='tpr', x.measure='fpr')

plot(SVMroc, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0),
     main = "ROC for SVM Model")
lines(x=c(0,1), y=c(0,1), col='red')
```



```r
SVMauc <- ROCR::performance(SVMdob, measure = "auc")
print(SVMauc@y.values)
```

```
## [[1]]
## [1] 0.9997814
```

Training Data Performance Table

```r
PerfTable <- data.frame(rbind(KNNPerf, LDAPerf, QDAPerf, LogPerf, PLogPerf,
                              RFPerf, SVMPerf))

colnames(PerfTable) <- c("Model", "Tuning", "AUROC", "Threshold",
                                "Accuracy", "TPR", "FPR", "Precision")

rownames(PerfTable) <- NULL
```

```r
knitr::kable(PerfTable, "simple")
```

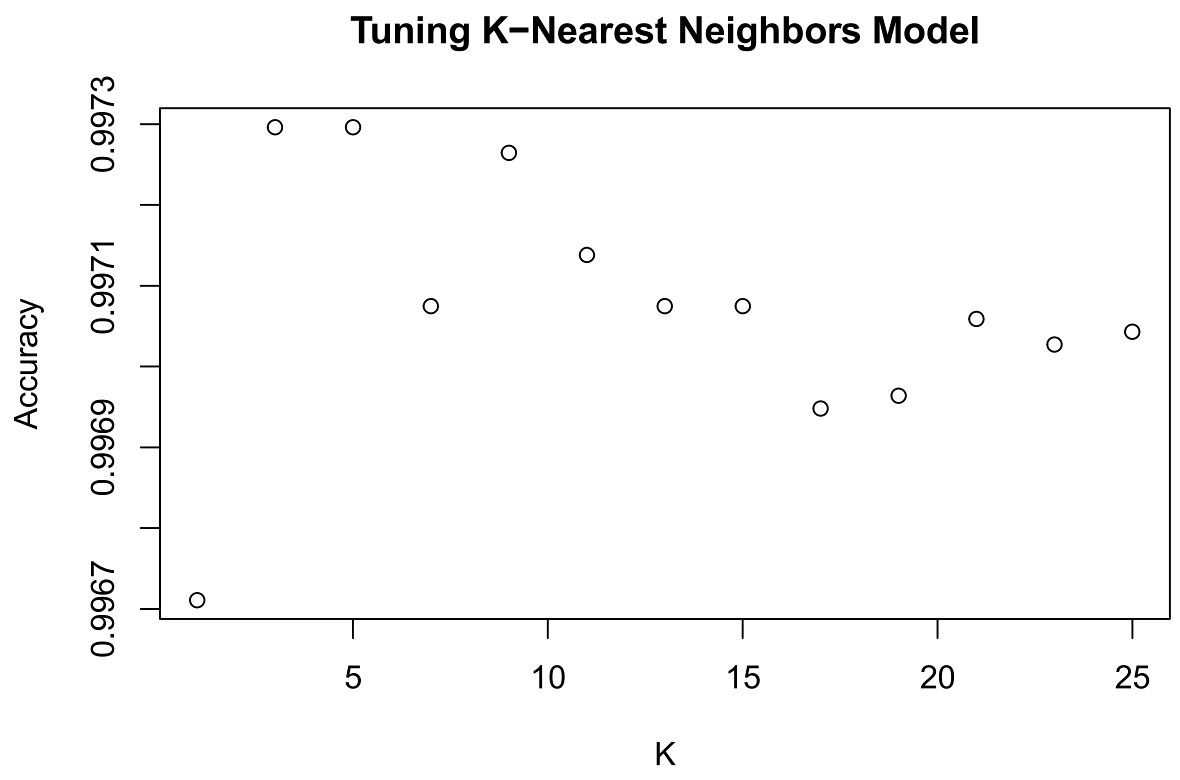
Model	Tuning	AUROC	Threshold	Accuracy	TPR	FPR	Precision
K-Nearest Neighbors	K = 5	0.9999		0.9979	0.9989	0.0321	0.9989
Linear Discriminant Analysis		0.9889	0.5	0.984	0.99	0.1988	0.9934
Quadratic Discriminant Analysis		0.9982	0.5	0.9946	0.9997	0.1597	0.9948
Logistic Regression		0.9985	0.5	0.9953	0.9989	0.1147	0.9934
Penalized Logistic Regression	Lambda = 0.001	0.9971	0.5	0.9921	1	0.2468	0.9919
Random Forest	Mtry = 1, Trees = 100	0.9999	0.5	0.9994	0.9999	0.0143	0.9995
Support Vector Machine	Radial, C = 100, Sigma = 3	0.9998	0.5	0.9975	0.9988	0.0415	0.9986

Most of our models delivered very high overall accuracy on the cross-validated training set. However, we should remember that the null model was 96.8% accurate. Of the seven models we analyzed, the best accuracy was random forest at 99.9%. A close second was KNN at 99.8%. All models exceeded the no-information rate of 96.8% accurate.

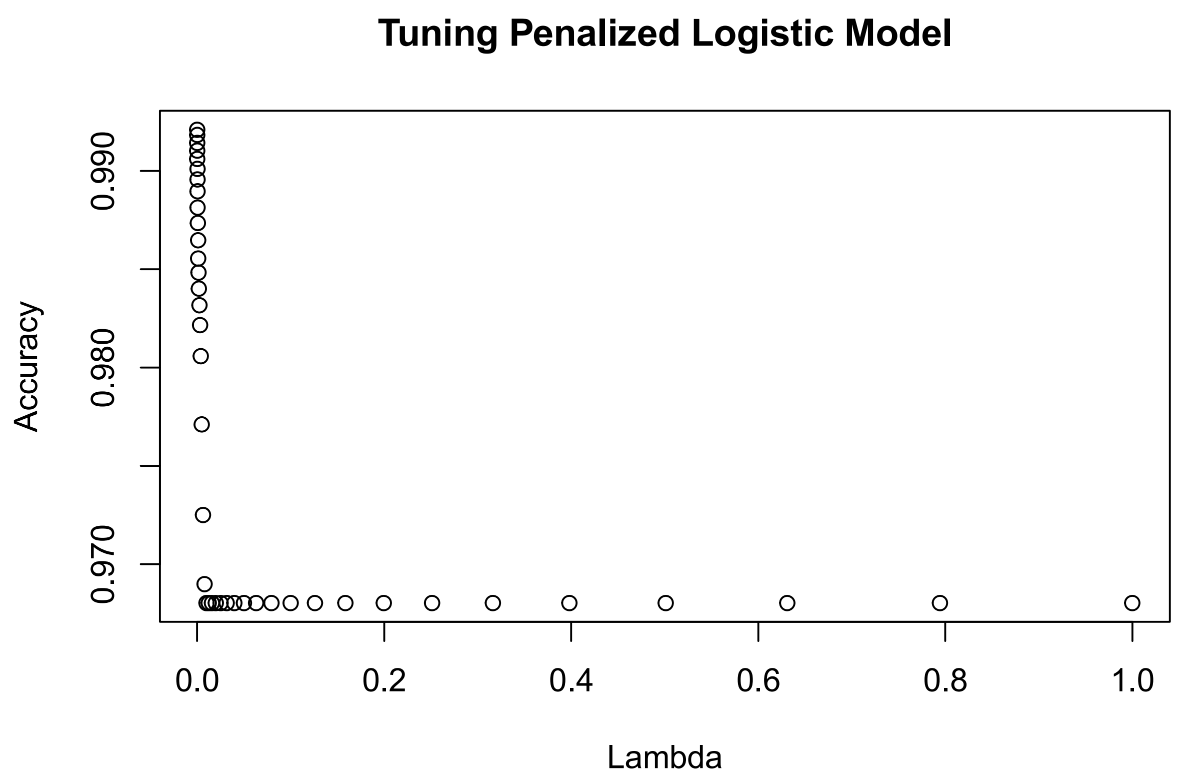
Tuning Parameters

Four of our models required tuning - KNN, penalized logistic regression, random forest, and SVM. In all model types, we tried a range of tuning values, then used cross-validation to select the one with the highest accuracy. In the case of KNN, k=5 had the highest accuracy, so it was chosen as the tuning value. In the case of penalized logistic regression, the highest accuracy occurred when lambda was 0.0001. For the random forest, the highest accuracy was an mtry value of 1. Finally, the SVM model saw highest accuracy with a cost of 100, sigma of 3, and a radial kernel.

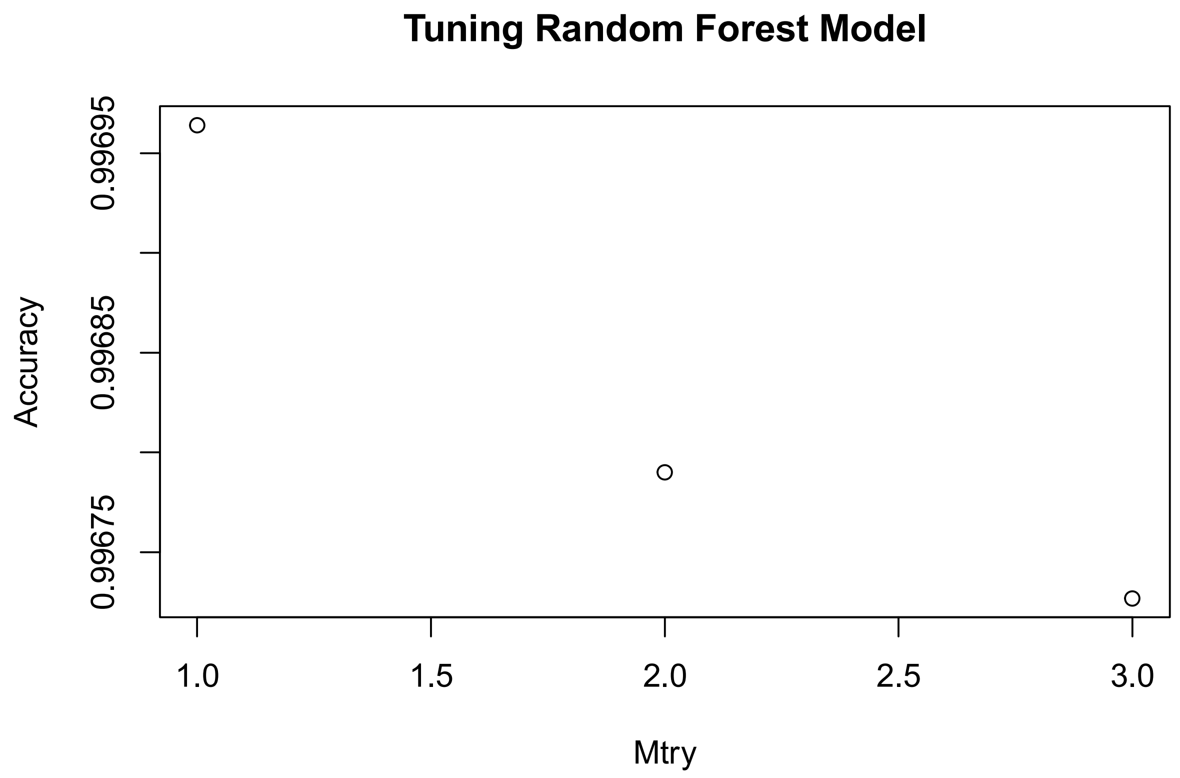
```
plot(x = KNNModel$results$k, y = KNNModel$results$Accuracy, xlab = "K",
     ylab = "Accuracy", main = "Tuning K-Nearest Neighbors Model")
```



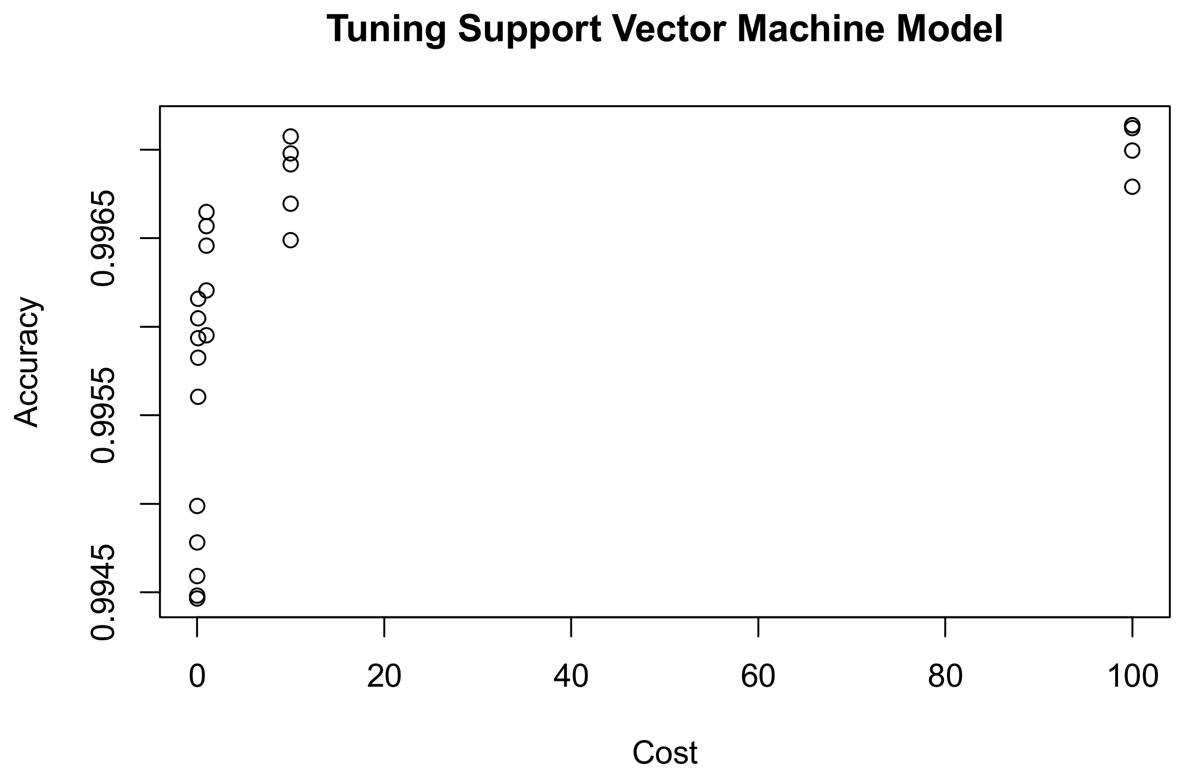
```
plot(x = PLogModel$results$lambda, y = PLogModel$results$Accuracy,
     xlab = "Lambda", ylab = "Accuracy",
     main = "Tuning Penalized Logistic Model")
```



```
plot(x = RFModel1$results$mtry, y = RFModel1$results$Accuracy,
     xlab = "Mtry", ylab = "Accuracy",
     main = "Tuning Random Forest Model")
```



```r
plot(x = SVMModelR$results$C, y = SVMModelR$results$Accuracy,
     xlab = "Cost", ylab = "Accuracy",
     main = "Tuning Support Vector Machine Model")
```



Threshold Selection

For every model except KNN, we used outputs in terms of probability percentages. Thus, most of the models needed a prediction threshold to say whether a given pixel was a blue tarp or not. KNN did not require a threshold since it directly outputs class predictions.

The default threshold is 50%, assuming that a false positive is equally problematic as a false negative. In this case, we believe such an assumption is reasonable. While we do not want to miss people in need of help, we also do not want to squander resources in areas that do not have people in need. As our relief resources are likely finite, both of these outcomes would lead to suffering and/or death. If we had a better sense of the scope of our resources, we could fine-tune the threshold. For now, we use 50%.

Test Data Wrangling

```r
T1 <- read_table2("orthovnir067_ROI_Blue_Tarps.txt")

names(T1)[8] <- "Red"
names(T1)[9] <- "Green"
names(T1)[10] <- "Blue"

T1 <- subset(T1, select = c("Red", "Green", "Blue"))

T1 <- mutate(T1, BlueTarp = 1)
T1$BlueTarp <- as.factor(T1$BlueTarp)
```

```r
T2 <- read_table2("orthovnir069_ROI_Blue_Tarps.txt")

names(T2)[8] <- "Red"
names(T2)[9] <- "Green"
names(T2)[10] <- "Blue"

T2 <- subset(T2, select = c("Red", "Green", "Blue"))

T2 <- mutate(T2, BlueTarp = 1)
T2$BlueTarp <- as.factor(T2$BlueTarp)
```

```r
T3 <- read_table2("orthovnir078_ROI_Blue_Tarps.txt")

names(T3)[8] <- "Red"
names(T3)[9] <- "Green"
names(T3)[10] <- "Blue"

T3 <- subset(T3, select = c("Red", "Green", "Blue"))

T3 <- mutate(T3, BlueTarp = 1)
T3$BlueTarp <- as.factor(T3$BlueTarp)
```

```r
T4 <- read_table2("orthovnir057_ROI_NON_Blue_Tarps.txt")

names(T4)[8] <- "Red"
names(T4)[9] <- "Green"
names(T4)[10] <- "Blue"

T4 <- subset(T4, select = c("Red", "Green", "Blue"))

T4 <- mutate(T4, BlueTarp = 0)
T4$BlueTarp <- as.factor(T4$BlueTarp)
```

```r
T5 <- read_table2("orthovnir067_ROI_NOT_Blue_Tarps.txt")

names(T5)[8] <- "Red"
names(T5)[9] <- "Green"
names(T5)[10] <- "Blue"

T5 <- subset(T5, select = c("Red", "Green", "Blue"))

T5 <- mutate(T5, BlueTarp = 0)
T5$BlueTarp <- as.factor(T5$BlueTarp)
```

```r
T6 <- read_table2("orthovnir069_ROI_NOT_Blue_Tarps.txt")

names(T6)[8] <- "Red"
names(T6)[9] <- "Green"
names(T6)[10] <- "Blue"

T6 <- subset(T6, select = c("Red", "Green", "Blue"))

T6 <- mutate(T6, BlueTarp = 0)
T6$BlueTarp <- as.factor(T6$BlueTarp)
```

```r
T7 <- read_table2("orthovnir078_ROI_NON_Blue_Tarps.txt")

names(T7)[8] <- "Red"
names(T7)[9] <- "Green"
names(T7)[10] <- "Blue"
```

```
T7 <- subset(T7, select = c("Red", "Green", "Blue"))

T7 <- mutate(T7, BlueTarp = 0)
T7$BlueTarp <- as.factor(T7$BlueTarp)
```

```
Haiti_Test <- data.frame(rbind(T1, T2, T3, T4, T5, T6, T7))
Test_Real <- Haiti_Test$BlueTarp
```

K-Nearest Neighbors Test

```
set.seed(1)
KNNPred_Test <- predict(KNNModel, newdata = Haiti_Test)
KNNOutput_Test <- caret::confusionMatrix(data = KNNPred_Test, reference = Test_Real)
KNNOutput_Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       1       0
##          1   11847   12772
##          0    2633 1976925
##
##                Accuracy : 0.9923
##                  95% CI : (0.9922, 0.9924)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.6024
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.818163
##             Specificity : 0.993581
##          Pos Pred Value : 0.481214
##          Neg Pred Value : 0.998670
##              Prevalence : 0.007225
##          Detection Rate : 0.005911
##    Detection Prevalence : 0.012284
##       Balanced Accuracy : 0.905872
##
##        'Positive' Class : 1
##
```

```
Test_Real2 <- as.numeric(Test_Real)
KNNPred_Test2 <- predict(KNNModel, newdata = Haiti_Test, type = 'prob')
KNNdob_Test <- ROCR::prediction(KNNPred_Test2['0'], Test_Real2)

KNNauc_Test <- ROCR::performance(KNNdob_Test, measure = "auc")
print(KNNauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9494872
```

```
KNNPerf_Test <- c("K-Nearest Neighbors", "K = 5", 0.9495, "", 0.9923, 0.8182,
                  round(1 - 0.9936, 4), 0.4812)
```

Linear Discriminant Analysis Test

```
LDAPred_Test <- predict(LDAModel, newdata = Haiti_Test)
LDAOutput_Test <- caret::confusionMatrix(data = LDAPred_Test, reference = Test_Real)
LDAOutput_Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       1       0
##          1   12148   34245
##          0    2332 1955452
##
```

```
##                Accuracy : 0.9817
##                  95% CI : (0.9816, 0.9819)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3924
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.838950
##             Specificity : 0.982789
##          Pos Pred Value : 0.261850
##          Neg Pred Value : 0.998809
##              Prevalence : 0.007225
##          Detection Rate : 0.006061
##    Detection Prevalence : 0.023148
##       Balanced Accuracy : 0.910870
##
##        'Positive' Class : 1
##
```

```
LDAPred_Test2 <- predict(LDAModel, newdata = Haiti_Test, type = 'prob')
LDAdob_Test <- ROCR::prediction(LDAPred_Test2['0'], Test_Real2)

LDAauc_Test <- ROCR::performance(LDAdob_Test, measure = "auc")
print(LDAauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9921155
```

```
LDAPerf_Test <- c("Linear Discriminant Analysis", "", 0.9921, 0.50, 0.9817, 0.8390,
                  round(1 - 0.9828, 4), 0.2619)
```

Quadratic Discriminant Analysis Test

```
QDAPred_Test <- predict(QDAModel, newdata = Haiti_Test)
QDAOutput_Test <- caret::confusionMatrix(data = QDAPred_Test, reference = Test_Real)
QDAOutput_Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       1       0
##          1   10058    3651
##          0    4422 1986046
##
##                Accuracy : 0.996
##                  95% CI : (0.9959, 0.9961)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7116
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.694613
##             Specificity : 0.998165
##          Pos Pred Value : 0.733679
##          Neg Pred Value : 0.997778
##              Prevalence : 0.007225
##          Detection Rate : 0.005019
##    Detection Prevalence : 0.006840
##       Balanced Accuracy : 0.846389
##
##        'Positive' Class : 1
##
```

```
QDAPred_Test2 <- predict(QDAModel, newdata = Haiti_Test, type = 'prob')
QDAdob_Test <- ROCR::prediction(QDAPred_Test2['1'], Test_Real)

QDAauc_Test <- ROCR::performance(QDAdob_Test, measure = "auc")
print(QDAauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9915001
```

```
QDAPerf_Test <- c("Quadratic Discriminant Analysis", "", 0.9915, 0.50, 0.9960, 0.6946,
                  round(1 - 0.9982, 4), 0.7337)
```

Logistic Regression Test

```
LogPred_Test <- predict(LogModel, newdata = Haiti_Test)
LogOutput_Test <- caret::confusionMatrix(data = LogPred_Test, reference = Test_Real)
LogOutput_Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       1       0
##          1   14310   20314
##          0     170 1969383
##
##                Accuracy : 0.9898
##                  95% CI : (0.9896, 0.9899)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.5786
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.988260
##             Specificity : 0.989790
##          Pos Pred Value : 0.413297
##          Neg Pred Value : 0.999914
##              Prevalence : 0.007225
##          Detection Rate : 0.007140
##    Detection Prevalence : 0.017276
##       Balanced Accuracy : 0.989025
##
##        'Positive' Class : 1
##
```

```
LogPred_Test2 <- predict(LogModel, newdata = Haiti_Test, type = 'prob')
Logdob_Test <- ROCR::prediction(LogPred_Test2['1'], Test_Real)

Logauc_Test <- ROCR::performance(Logdob_Test, measure = "auc")
print(Logauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9994131
```

```
LogPerf_Test <- c("Logistic Regression", "", 0.9994, 0.50, 0.9898, 0.9883,
                  round(1 - 0.9898, 4), 0.4133)
```

Penalized Logistic Regression Test

```
Haiti_Test2 <- Haiti_Test
levels(Haiti_Test2$BlueTarp) <- c('linked', 'unrelated')

Test_Real3 <- Test_Real
levels(Test_Real3) <- c('linked', 'unrelated')

PLogPred_Test <- predict(PLogModel, newdata = Haiti_Test2)
PLogOutput_Test <- caret::confusionMatrix(data = PLogPred_Test, reference = Test_Real3)
PLogOutput_Test
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   linked unrelated
##   linked      12747      1592
##   unrelated    1733   1988105
```

```
##
##                Accuracy : 0.9983
##                  95% CI : (0.9983, 0.9984)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8838
##
##  Mcnemar's Test P-Value : 0.01519
##
##             Sensitivity : 0.880318
##             Specificity : 0.999200
##          Pos Pred Value : 0.888974
##          Neg Pred Value : 0.999129
##              Prevalence : 0.007225
##          Detection Rate : 0.006360
##    Detection Prevalence : 0.007155
##       Balanced Accuracy : 0.939759
##
##        'Positive' Class : linked
##
```

```
PLogPred_Test2 <- predict(PLogModel, newdata = Haiti_Test2, type = 'prob')
PLogdob_Test <- ROCR::prediction(PLogPred_Test2[1], Test_Real3)

PLogauc_Test <- ROCR::performance(PLogdob_Test, measure = "auc")
print(PLogauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9986075
```

```
PLogPerf_Test <- c("Penalized Logistic Regression", "Lambda = 0.001", 0.9986, 0.50, 0.9983,
              0.8803, round(1 - 0.9992, 4) , 0.8890)
```

Random Forest Test

```
set.seed(1)
RFPred_Test <- predict(RFModel1, newdata = Haiti_Test)
RFOutput_Test <- caret::confusionMatrix(data = RFPred_Test, reference = Test_Real)
RFOutput_Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       1       0
##          1   11255    7471
##          0    3225 1982226
##
##                Accuracy : 0.9947
##                  95% CI : (0.9946, 0.9948)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6752
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.777279
##             Specificity : 0.996245
##          Pos Pred Value : 0.601036
##          Neg Pred Value : 0.998376
##              Prevalence : 0.007225
##          Detection Rate : 0.005616
##    Detection Prevalence : 0.009343
##       Balanced Accuracy : 0.886762
##
##        'Positive' Class : 1
##
```

```r
RFPred_Test2 <- predict(RFModel1, newdata = Haiti_Test, type = 'prob')
RFdob_Test <- ROCR::prediction(RFPred_Test2['1'], Test_Real)

RFauc_Test <- ROCR::performance(RFdob_Test, measure = "auc")
print(RFauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9825966
```

```r
RFPerf_Test <- c("Random Forest", "Mtry = 1, Trees = 100", 0.9999, 0.50, 0.9926, 0.7773,
             round(1 - 0.9962, 4), 0.6010)
```

Support Vector Machines Test

```r
set.seed(1)
SVMPred_Test <- predict(SVMModelR, newdata = Haiti_Test)
SVMOutput_Test <- caret::confusionMatrix(data = SVMPred_Test, reference = Test_Real3)
SVMOutput_Test
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   linked unrelated
##   linked       6837     14457
##   unrelated    7643   1975240
##
##                Accuracy : 0.989
##                  95% CI : (0.9888, 0.9891)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3769
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 0.472169
##             Specificity : 0.992734
##          Pos Pred Value : 0.321076
##          Neg Pred Value : 0.996146
##              Prevalence : 0.007225
##          Detection Rate : 0.003411
##    Detection Prevalence : 0.010625
##       Balanced Accuracy : 0.732451
##
##        'Positive' Class : linked
##
```

```
SVMPred_Test2 <- predict(SVMModelR, newdata = Haiti_Test, type = 'prob')
SVMdob_Test <- ROCR::prediction(SVMPred_Test2['unrelated'], Test_Real2)

SVMauc_Test <- ROCR::performance(SVMdob_Test, measure = "auc")
print(SVMauc_Test@y.values)
```

```
## [[1]]
## [1] 0.9529235
```

```
SVMPerf_Test <- c("Support Vector Machine", "Radial, C = 100, Sigma = 3", 0.9529,
             0.50, 0.9890, 0.4722, round(1 - 0.9927, 4), 0.3211)
```

Test Data Performance Table

```
PerfTable_Test <- data.frame(rbind(KNNPerf_Test, LDAPerf_Test, QDAPerf_Test,
                                   LogPerf_Test, PLogPerf_Test,
                                    RFPerf_Test, SVMPerf_Test))

colnames(PerfTable_Test) <- c("Model", "Tuning", "AUROC", "Threshold",
                                      "Accuracy", "TPR", "FPR", "Precision")
rownames(PerfTable_Test) <- NULL

knitr::kable(PerfTable_Test, "simple")
```

Model	Tuning	AUROC	Threshold	Accuracy	TPR	FPR	Precision
K-Nearest Neighbors	K = 5	0.9495		0.9923	0.8182	0.0064	0.4812
Linear Discriminant Analysis		0.9921	0.5	0.9817	0.839	0.0172	0.2619
Quadratic Discriminant Analysis		0.9915	0.5	0.996	0.6946	0.0018	0.7337
Logistic Regression		0.9994	0.5	0.9898	0.9883	0.0102	0.4133
Penalized Logistic Regression	Lambda = 0.001	0.9986	0.5	0.9983	0.8803	8e-04	0.889
Random Forest	Mtry = 1, Trees = 100	0.9999	0.5	0.9926	0.7773	0.0038	0.601
Support Vector Machine	Radial, C = 100, Sigma = 3	0.9529	0.5	0.989	0.4722	0.0073	0.3211

Conclusions

Our first conclusion is about model accuracy. On the training data, the K-Nearest Neighbors model appears to be the best. It achieves the highest overall accuracy at 99.8%, and it balances the risk of Type I and Type II errors (see "Threshold Selection" section for why this matters). The KNN model also has the highest precision and the lowest false positive rate, while coming a close second in true positive rate. However, on the test data, which is what really matters, the best performing model is penalized logistic regression. Penalized logistic achieves the top test accuracy at 99.83%; it also has the lowest FPR and highest precision, as well as the second-highest TPR (after regular logistic regression).

Our second conclusion concerns the large difference in model performance ranking between training data and test data. On the training data, penalized logistic regression is not one of the top-performing models, but this completely flips in the test data. Similarly, KNN goes from having the best accuracy on the training data to being middle of the pack on the test data. And while all models beat the no-information accuracy on the training data, only 2 models did so on the test data: QDA and penalized logistic. This does not necessarily mean the other models are worse than no-information, since picking "not blue tarp" every time would not identify any of the blue tarps! However, the overall lack of accuracy on the test data for so many of the models is still disappointing. While it may be disappointing, it is not surprising. Even with cross-validation, training statistics are only an estimate of test data results. We performed extensive tuning on the training data, and likely overfit some of the models in the process. This is why an independent (or hold-out) data set is so important to evaluating model performance.

The third conclusion is our recommendation to use penalized logistic regression in the search for blue tarps. Penalized logistic regression comes in first or second on all of the key metrics in our performance table, meaning it is both very good at finding blue tarps and very good at avoiding pixels that are not blue tarps. Since both of these considerations are very important (see "Threshold Selection" section), we recommend penalized logistic. It also helps that this model is fast to train and fast to run on new data points, meaning it will prove easy and flexible to use in the real world.

The fourth conclusion is the importance of the the various model metrics we calculate for both the training and test data. The simplest measure is accuracy, the percentage of data points that our model correctly classifies (both blue tarps and not blue tarps). As mentioned earlier, penalized logistic regression has the highest accuracy on the test set. True positive rate is the percentage of real blue tarps that our model identifies as blue tarps. Clearly, this is vitally important in delivering aid to people in need. Penalized logistic regression comes in second on this metric in the test data, correctly identifying 88% of the blue tarps. False positive rate is the percentage of not blue tarps that our model incorrectly identifies as blue tarps. The higher this number, the more time we will waste sending aid to areas that likely do not have people. Almost all of our models minimize this number as a percent, but penalized logistic regression does the best on the test data. Next is the area under the ROC curve, which plots the

TPR/FPR trade-off. In an ideal model, there is no trade-off, AUROC is 1.0, and the ROC curve hugs the top left corner of the graph. Random forest nearly achieves this, and both logistic regression and penalized logistic regression are close behind. We also calculate precision, the fraction of blue tarp predictions that actually turn out to be blue tarps. The higher this number, the more "precisely" we can target people in need. Penalized logistic regression is the clear winner on precision in the test data.

Our fifth conclusion is the problem of imbalanced data sets and what that means for model construction. In the training data, only 3.2% of observations were blue tarps, and in the test data, the number was only 0.7%. Thus, our training models saw mostly not blue tarps, and when looking at the test data, the models had to parse out a tiny fraction of the total data to correctly identify as blue tarps. One possible solution would be random under-sampling of not blue tarps to reduce their ratio versus blue tarps. Alternately, we could randomly over-sample blue tarps. A second possible solution is using class-specific penalization in our models. For example, in a loss function, we could assign greater weight to misclassifying a blue tarp as opposed to not blue tarps.

Our sixth and final conclusion is about how to use the final model. Since we do not have a good sense of the resources available to conduct our search for displaced people, we may not be able to search all of the areas identified as blue tarps by the penalized logistic regression model. As a result, we may wish to use probabilities instead of binary Yes/No classification. If we only have the resources to search 1000 areas (for example), we can search the 1000 with the highest probability of being blue tarps.