Disaster Relief Project Part 1

Sirish

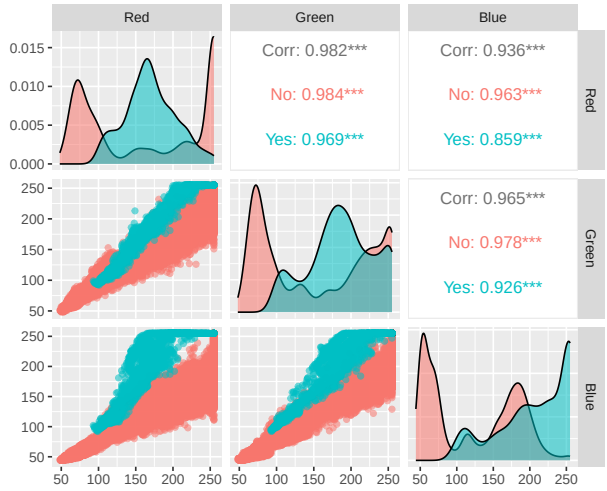
2023-03-08

```
library(ISLR)
library(boot)
library(caret)
library(tidyverse)
library(broom)
library(knitr)
library(kableExtra)
library(leaps)
library(gam)
library(ggplot2)
library(GGally)
library(ROCR)
```

```
data <- read.csv("HaitiPixels.csv")
```

```
data$Tarp <- ifelse(data$Class == "Blue Tarp", "Yes", "No")
data$Tarp <- factor(data$Tarp, levels = c("No", "Yes"))
```

```
data %>%
  dplyr::select(-Class, -Tarp) %>%
  ggpairs(upper=list(continuous=wrap('cor', size=4)),
          mapping=ggplot2::aes(color=data$Tarp, alpha=0.5))
```



As Blue Tarp pixels decrease in size, the values decreases, which could be correlated to the pixels

getting darker as they are smaller on a map.
These graphs also show that it is unlikely to find a Blue Tarp with a high Red value, which
I believe to be because Red and Blue aren't very similar as colors. They also have different values
on the RGB scale, which could also relate to Red and Blue being different. While I do think its possible
to have a Blue Tarp with a high green value because green and blue are similar in RGB values.
Therefore, it may be important to check the interaction between green and blue.

```
seed <- 1
folds <- createFolds(data$Tarp, k=10, list=T, returnTrain=T)
```

```
control <- trainControl(method = 'cv',
                        number=10,
                        index=folds,
                        savePredictions=T,
                        classProbs=T)
```

```
stats <- c("Accuracy", "Sensitivity", "Specificity", "Precision")

threshold_test <- function(model) {
  set.seed(seed)
  stats <- thresholder(model,
                       threshold=seq(0.1,0.9, by=0.1),
                       statistics=stats)
  stats$FalseNegative <- 1-stats$Sensitivity
  stats$FalsePositive <- 1-stats$Specificity
  return(stats)
}
```

```
plotROC <- function(model, stated, model_name){
  set.seed(seed)
  prob <- model$pred[order(model$pred$rowIndex),]
  rates<- prediction(prob$Yes, as.numeric(data$Tarp))
  roc <- performance(rates, measure='tpr', x.measure='fpr')
  plot(roc, main=paste("ROC Curve:", model_name))
  lines(x=c(0,1), y=c(0,1), col='red')

  auc<- performance(rates,"auc")
  stated <- stated %>% mutate(AUROC = auc@y.values[[1]])
  return(stated)
}
```

Logistic Regression

```
set.seed(seed)
datalog <- train(Tarp~Red+Green+Blue,
                 data=data,
                 family='binomial',
                 method='glm',
                 trControl=control)
datalog$results
```

```
#>   parameter Accuracy     Kappa AccuracySD    KappaSD
#> 1      none 0.995272 0.9202367 0.00087395 0.01576765
```

```
datalogwinteraction <- train(Tarp~Red+Green+Blue+Green:Blue,
                  data=data,
                  family='binomial',
                  method='glm',
                  trControl=control)
datalogwinteraction$results
```

```
#>   parameter  Accuracy     Kappa   AccuracySD    KappaSD
#> 1      none 0.9954776 0.9252266 0.0007680991 0.01322292
```

After checking the accuracy of the using interaction and without, it does not seem beneficial to justify adding the Green:Blue interaction term.

```
datalog.thresh<- threshold_test(datalog)
datalog.thresh[2:8] %>% slice(which.max(Accuracy))
```

```
#>   prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1            0.7 0.9956041   0.9984319   0.9099888 0.9970319   0.001568147
#>   FalsePositive
#> 1    0.09001122
```

```
dataloginteraction.thresh<- threshold_test(datalogwinteraction)
dataloginteraction.thresh[2:8] %>% slice(which.max(Accuracy))
```

```
#>   prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1            0.8 0.9960627   0.9972067   0.9614325 0.9987242   0.002793258
#>   FalsePositive
#> 1    0.03856753
```

The highest accuracy threhold for no-interaction was 0.7, while that the interaction model was 0.7. The accuracy of the interaction threshold was higher, however since the False Negative Rate is lower of the noninteraction was lower and the decrease in accuracy was insignificant enough that we can just go with the noninteraction model.

```
logselected <-datalog.thresh[2:8] %>% slice(which.max(Accuracy))
logselected <-plotROC(datalog, logselected, "Logistic Regression")
```

```
logselected[["AUROC"]]
```

```
#> [1] 0.9984586
```

The ROC curve for the selected logistic regression model was very good.
The True positive rate reaches 1 barely above a false positive rate of 0.
The AUROC is 0.9984746.

Linear Discriminant Analysis

```
set.seed(seed)
datalda <- train(Tarp~Red+Green+Blue,
                 data=data,
                 method='lda',
                 trControl=control)

dataldawinteraction <- train(Tarp~Red+Green+Blue+Green:Blue,
                 data=data,
                 method='lda',
                 trControl=control)
datalda$results %>% slice(which.max(Accuracy))
```

```
#>   parameter  Accuracy     Kappa  AccuracySD    KappaSD
#> 1      none 0.9838396 0.7520677 0.001554598 0.02006839
```

```
dataldawinteraction$results %>% slice(which.max(Accuracy))
```

```
#>   parameter  Accuracy     Kappa   AccuracySD    KappaSD
#> 1      none 0.9942759 0.9018961 0.0007881821 0.01452504
```

There was a significant enough difference in accuracies between the interaction of Green and Blue to justify using the interaction model. There was also a lower accuracy standard deviation which makes me think that the interaction term is more important and we can trust the information we get from it more.

```
datalda.thresh <- threshold_test(datalda)
datalda.thresh[2:8] %>% slice(which.max(Accuracy))
```

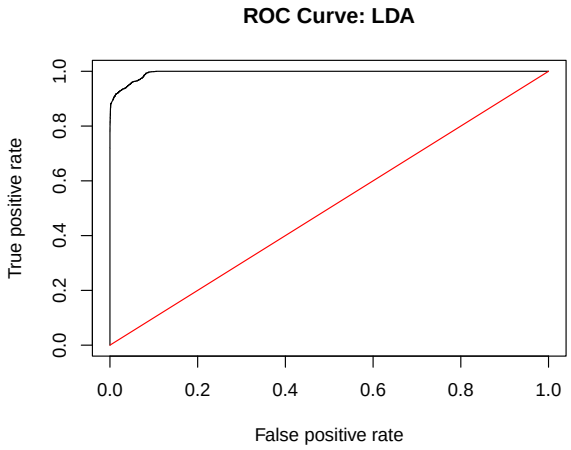
```
#>   prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1            0.1 0.9846144   0.9926493   0.7413208 0.9914679   0.007350672
#>   FalsePositive
#> 1     0.2586792
```

```
dataldawinteraction.thresh <- threshold_test(dataldawinteraction)
dataldawinteraction.thresh[2:8] %>% slice(which.max(Accuracy))
```

```
#>   prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1            0.7 0.9944814   0.9987096    0.866461 0.9956037   0.001290447
#>   FalsePositive
#> 1      0.133539
```

The highest accuracy threshold for the non-interaction term was 0.1, while using the interaction model it was 0.7. The interaction model also had a higher Accuracy, Sensitivity, Specificity, and Precision

```
ldaselect <- dataldawinteraction.thresh[2:8] %>% slice(which.max(Accuracy))
ldaselect <- plotROC(dataldawinteraction, ldaselect, "LDA")
```




```
ldaselect[['AUROC']]
```

```
#> [1] 0.9952365
```

The ROC curve isn't as sharp as the logistic regression model's ROC curve.
There is a sharp increase at the True Positive Rate of 0.8 and a dropoff after that.
The AUROC for this model is 0.9952359.

Quadratic Discriminant Analysis (QDA)

```
set.seed(seed)
dataqda <- train(Tarp~Red+Green+Blue,
                 data=data,
                 method='qda',
                 trControl=control)

dataqdawinteraction <- train(Tarp~Red+Green+Blue+Green:Blue,
                 data=data,
                 method='qda',
                 trControl=control)
dataqda$results %>% slice(which.max(Accuracy))
```

```
#>   parameter  Accuracy    Kappa  AccuracySD    KappaSD
#> 1      none 0.9945763 0.905267 0.000850188 0.01589921
```

```
dataqdawinteraction$results %>% slice(which.max(Accuracy))
```

```
#>   parameter  Accuracy     Kappa  AccuracySD    KappaSD
#> 1      none 0.9890419 0.8362136 0.000971932 0.01261794
```

The non-interaction model had a higher accuracy and smaller accuracy standard deviation than the interaction model.

```
dataqda.thresh <- threshold_test(dataqda)
dataqda.thresh[2:8] %>% slice(which.max(Accuracy))
```

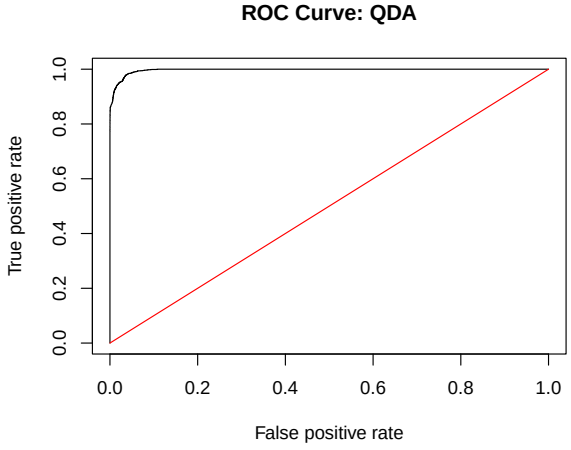
```
#>   prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1            0.7 0.9947344   0.9992976   0.8565673 0.9952825  0.0007023955
#>   FalsePositive
#> 1     0.1434327
```

```
dataqdawinteraction.thresh <- threshold_test(dataqdawinteraction)
dataqdawinteraction.thresh[2:8] %>% slice(which.max(Accuracy))
```

```
#>   prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1            0.1 0.9938805   0.9981379   0.8649783  0.995554   0.001862147
#>   FalsePositive
#> 1     0.1350217
```

The highest accuracy threshold for no interaction model was 0.8, while the interaction model had 0.1. The non-interaction model also had higher values for every metric.

```
qdaselect <- dataqda.thresh[2:8] %>% slice(which.max(Accuracy))
qdaselect <-plotROC(dataqdawinteraction, qdaselect, "QDA")
```




```
qdaselect[['AUROC']]
```

```
#> [1] 0.9970429
```

The ROC had a similar curve to LDA, where there was a sharp increase at 0.8, however it was more smoother after 0.8. The AUROC for this model is 0.9970526

KNN

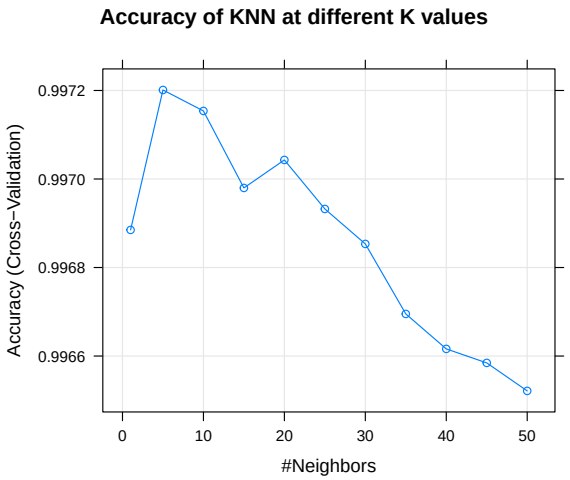
```
set.seed(seed)
ks <- data.frame(k=seq(0,50,5))
ks[1,] <-1

dataknn <- train(Tarp~Red+Green+Blue,
                 data=data,
                 tuneGrid=ks,
                 method='knn',
                 metric="Accuracy",
                 trControl=control)

dataknnwinteraction <- train(Tarp~Red+Green+Blue+Green:Blue,
                 data=data,
                 tuneGrid=ks,
                 method='knn',
                 metric="Accuracy",
                 trControl=control)
dataknn$results %>% slice(which.max(Accuracy))
```

```
#>   k  Accuracy     Kappa   AccuracySD     KappaSD
#> 1 5 0.9972012 0.9548958 0.0003339551 0.005430985
```

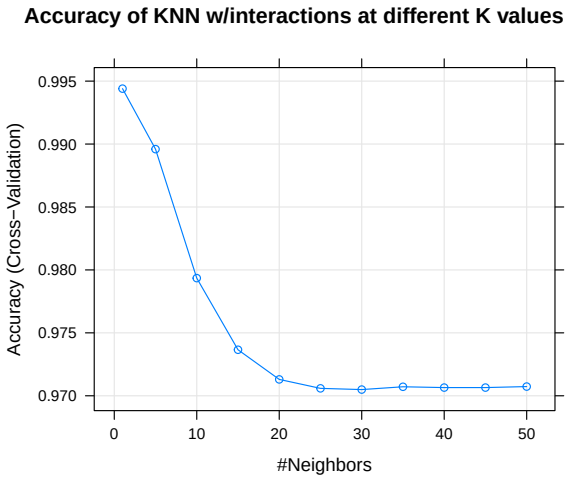
```
dataknnwinteraction$results %>% slice_max(Accuracy)
```

```
#>   k  Accuracy     Kappa   AccuracySD    KappaSD
#> 1 1 0.9944024 0.9076243 0.0008070999 0.01407622
```

```
plot(dataknn, main="Accuracy of KNN at different K values")
```




```
plot(dataknnwinteraction, main="Accuracy of KNN w/interactions at different K values")
```



The model shows that the highest accuracy is when K=5. However, at K=5, we have the chance for the model to be more biased towards the training dataset and not be able to be as accurate for the test data set. I am not going to use K=10, because I feel that with this large of a dataset, it will still run into the same issues. So, since we have a large enough dataset, I am deciding to use K=20, because its the largest K can be, while still being above 0.9970. Also the interaction dataset had a K-value of 1, and I don't think that would be useful when using a test data set.There is also a massive drop off in accuracy when adding more K values.

```
dataknn10 <- train(Tarp~Red+Green+Blue,
                   data=data,
                   tuneGrid= data.frame(k=seq(10,10,1)),
                   method='knn',
                   metric="Accuracy",
                   trControl=control)
dataknn20 <- train(Tarp~Red+Green+Blue,
                   data=data,
                   tuneGrid= data.frame(k=seq(20,20,1)),
                   method='knn',
                   metric="Accuracy",
                   trControl=control)
```

```
dataknn.thresh <-threshold_test(dataknn)
dataknn10.thresh <-threshold_test(dataknn10)
dataknn20.thresh <-threshold_test(dataknn20)

dataknn.thresh %>% slice(which.max(Accuracy))
```

```
#>   k prob_threshold Accuracy Sensitivity Specificity Precision FalseNegative
#> 1 5            0.4 0.997217   0.9985625   0.9564771 0.9985628   0.001437471
#>   FalsePositive
#> 1     0.0435229
```

```
dataknn10.thresh %>% slice(which.max(Accuracy))
```

```
#>    k prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
```

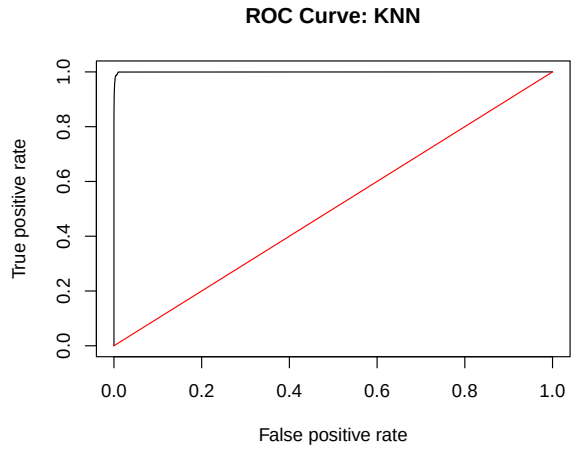
```
#> 1 10            0.4 0.9972644   0.9986605    0.954992 0.9985139   0.001339458
#>   FalsePositive
#> 1    0.04500805
```

```
dataknn20.thresh %>% slice(which.max(Accuracy))
```

```
#>    k prob_threshold  Accuracy Sensitivity Specificity Precision FalseNegative
#> 1 20            0.5 0.9970431   0.9982848   0.9594474 0.9986603   0.001715155
#>   FalsePositive
#> 1     0.0405526
```

When comparing k=5, k=10, and k=20 we can see that they all have very similar metrics overall, except Accuracy was slightly lower at K=20. Therefore, it is more useful to use K=20 because it will allow for more optimal model when using the test data set.

```
knnselect <- dataknn20.thresh %>% slice(which.max(Accuracy))
knnselect <- plotROC(dataknn20, knnselect, "KNN")
```




```
knnselect[['AUROC']]
```

```
#> [1] 0.9995057
```

The ROC curve for this KNN model is the best curve we have seen so far.
The AUROC for this curve is 0.999505.

Penalized Logistic Regression

```
set.seed(seed)

lambdaGrid <- expand.grid(alpha=0, lambda = seq(0,1,0.1))

dataridge <- train(Tarp~Red+Green+Blue,
                   data=data,
                   method='glmnet',
```

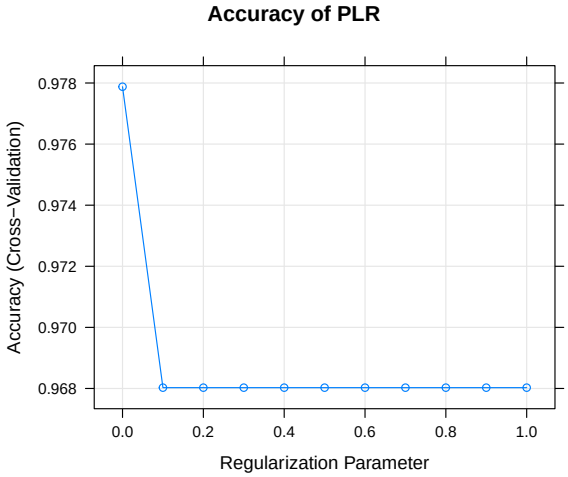
```
                  tuneGrid=lambdaGrid,
                  trControl=control)

dataridgewinteraction <- train(Tarp~Red+Green+Blue+Green:Blue,
                  data=data,
                  method='glmnet',
                  tuneGrid=lambdaGrid,
                  trControl=control)
dataridge$results %>% slice(which.max(Accuracy))
```

```
#>   alpha lambda  Accuracy     Kappa AccuracySD    KappaSD
#> 1     0      0 0.9778782 0.4609137 0.00159425 0.06088777
```

```
dataridgewinteraction$results %>% slice(which.max(Accuracy))
```

```
#>   alpha lambda  Accuracy     Kappa AccuracySD    KappaSD
#> 1     0      0 0.9784632 0.4822926 0.00149776 0.05498514
```

```
plot(dataridge, main="Accuracy of PLR")
```

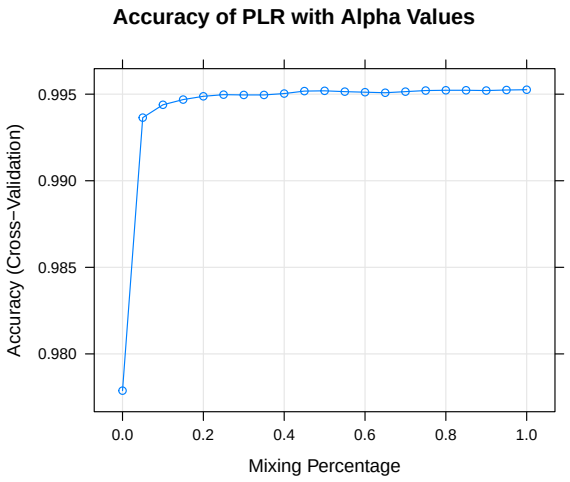


The optimal lambda value selected for the regression was 0.

```
alphaGrid <- expand.grid(alpha=seq(0,1,0.05), lambda = 0)

dataalpha <- train(Tarp~Red+Green+Blue,
                  data=data,
                  method='glmnet',
                  tuneGrid=alphaGrid,
                  trControl=control)

dataalpha$results %>% slice(which.max(Accuracy))
```

```
#>   alpha lambda  Accuracy    Kappa   AccuracySD    KappaSD
#> 1     1      0 0.9952562 0.919964 0.0008400746 0.01519755
```

```
plot(dataalpha, main="Accuracy of PLR with Alpha Values")
```



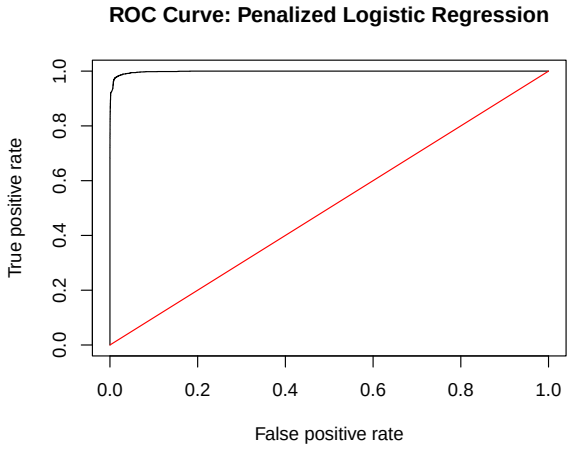
with a lambda of 0, the most accurate produced had an alpha of 0.95.
This accuracy was 0.9953037.
The optimal tuning parameters were alpha 0.95 and lambda 0.

```
dataalpha2 <- train(Tarp~Red+Green+Blue,
                    data=data,
                    method='glmnet',
                    tuneGrid=data.frame(alpha=.95, lambda=0),
                    trControl=control)
```

```
dataalpha.thresh <- threshold_test(dataalpha2)
dataalpha.thresh %>% slice(which.max(Accuracy))
```

```
#>   alpha lambda prob_threshold  Accuracy Sensitivity Specificity Precision
#> 1  0.95      0            0.7 0.9955883   0.9984972   0.9075135 0.9969508
#>   FalseNegative FalsePositive
#> 1   0.001502809    0.09248647
```

```
plrselect <- dataalpha.thresh %>% slice(which.max(Accuracy))
plrselect <- plotROC(dataalpha2, plrselect, "Penalized Logistic Regression")
```




```
plrselect[['AUROC']]
```

```
#> [1] 0.9984694
```

The ROC curve for the penalized logistic regression model looks good and similar to logistic regression. However, it is slightly more rounded like QDA curve.
The AUROC is 0.9984882

#Cross Validation

```
logselected <- logselected %>% mutate(Model="Log Reg")
ldaselect <- ldaselect %>% mutate(Model="LDA")
qdaselect <- qdaselect %>% mutate(Model="QDA")
knnselect <- knnselect %>% mutate(Model="KNN")
plrselect <- plrselect %>% mutate(Model="PLR")

cross <- data.frame(rbind(logselected,ldaselect,qdaselect))
cross<-cross %>%
  add_column(Tuning = NaN)
cross <- cross%>%
  add_column(Tuning2= NaN)

knnselect <- knnselect %>%
  add_column(Tuning2=NaN)
colnames(knnselect)[1] <-"Tuning"


plrselect <- plrselect[,c(12,1,2,3,4,5,6,7,8,9,10,11)]
```

```
#> Error in `[.data.frame`(plrselect, , c(12, 1, 2, 3, 4, 5, 6, 7, 8, 9, : undefined columns selected
```

```
colnames(plrselect)[1] <- "Tuning"
colnames(plrselect)[2] <- "Tuning2"

cross <- bind_rows(cross,plrselect)
```

```
cross <- bind_rows(cross,knnselect)
cross <-cross[,c(9,10,11,1,2,3,4,5,6,7,8)]
cross
```

```
#>     Model Tuning Tuning2 prob_threshold  Accuracy Sensitivity Specificity
#> 1 Log Reg    NaN     NaN            0.7 0.9956041   0.9984319   0.9099888
#> 2     LDA    NaN     NaN            0.7 0.9944814   0.9987096   0.8664610
#> 3     QDA    NaN     NaN            0.7 0.9947344   0.9992976   0.8565673
#> 4     PLR   0.95       0            0.7 0.9955883   0.9984972   0.9075135
#> 5     KNN  20.00     NaN            0.5 0.9970431   0.9982848   0.9594474
#>   Precision FalseNegative FalsePositive     AUROC
#> 1 0.9970319  0.0015681470    0.09001122 0.9984586
#> 2 0.9956037  0.0012904466    0.13353899 0.9952365
#> 3 0.9952825  0.0007023955    0.14343267 0.9970429
#> 4 0.9969508  0.0015028089    0.09248647 0.9984694
#> 5 0.9986603  0.0017151551    0.04055260 0.9995057
```

All of these solutions seemed to be useful in identifying whether something has blue tarps in the pixel based on RGB values, when training through 10-fold cross validations. The Logistic regression and penalized logistic regression were very similar which makes sense because the optimal lambda value was 0, so there were no predictors that were minimized to 0.

Looking at the chart itself, it looks like KNN k-value of 20 is still has the highest accuracy. It also looks like KNN also has the lowest False Positive rate, but does have the highest False Negative rate. Which was accounted for because I decided to use a higher K-value then was necessary to allow for an optimal model when it comes to the test data set.

These metrics were selected because when looking at the thresholds, I selected the one with the highest accuracy per model.

#Conclusion

Through the 5 models that were used, there was no clear winner. They all had very similar accuracies by only changing by at most .002 from the highest (KNN) to the lowest (LDA). The models LDA and QDA had by far the highest False Positive Rates, which is accompanied by a high sensitivity rate. Also had by far the lowest specificity, which is accompanied by a low false negative rate. Which shows that both models overclassify blue tarp pixels. There is also the issue of PLR having the highest AUC. This means that PLR would be the best at distinguishing between the positive and negative test classes. But that could also be due to a very low False Positive Rate.

In regards to the problem, having to identify blue tarp pixels, I believe that having a low False negative rate is really important. This is because we would like to overclassify what counts as blue because we would be able to pick-up more survivors in the area. Therefore, using QDA would be the most useful in this case. However, because the accuracy was still very high in all models, we are able to use any of them to some extent. However, since we care about trying to find blue tarp to help survivors, QDA would be what I use.

The KNN model had the highest accuracy and lowest false positive rate. Having a low false positive rate means that the search would be more efficient. This is because there will be some pixels that look close enough to a blue tarp pixel, therefore being able to already weed out what is a blue tarp pixel is a good thing. It is also worth nothing that the KNN model metric could be better if I used a lower K-value, however I wanted higher stability for the test data set.

Therefore, in a high risk situation where people's lives are at stake, I think having a low False negative rate is the most important thing. So, in juding using only that metric, the best model to use would be QDA. However, we also need to conserve resources in times of crisis to help the most amount of people so we woul dneed to be efficient with our searching; therefore, using that metric KNN would be the most efficient to not wasting time in searching for people. Therefore, in using both of these metrics, I think that using Logistic Regression would be the best. Even though, logistic regression has the the 4th highest False negative rate, it has the second lowest false positive rate. I think that this is a good compromise in not wasting too much resources when it comes to looking for people and being able to find the highest amount of people. It also has the 3rd highest AUC, and barely misses out on being 2nd. This means that it is decently good at being able to classify observations into their respective classes. However, I do not feel confident about this answer because the False Negative rate is so high.

#Appendix **code used to make the cross validation chart**

```
logselected <- logselected %>% mutate(Model="Log Reg")
ldaselect <- ldaselect %>% mutate(Model="LDA")
qdaselect <- qdaselect %>% mutate(Model="QDA")
knnselect <- knnselect %>% mutate(Model="KNN")
plrselect <- plrselect %>% mutate(Model="PLR")

cross <- data.frame(rbind(logselected,ldaselect,qdaselect))
cross<-cross %>%
  add_column(Tuning = NaN)
cross <- cross%>%
  add_column(Tuning2= NaN)

knnselect <- knnselect %>%
  add_column(Tuning2=NaN)
colnames(knnselect)[1] <-"Tuning"



#plrselect <- plrselect[,c(12,1,2,3,4,5,6,7,8,9,10,11)]
colnames(plrselect)[1] <- "Tuning"
colnames(plrselect)[2] <- "Tuning2"

#cross <- bind_rows(cross,plrselect)
#cross <- bind_rows(cross,knnselect)
cross <-cross[,c(9,10,11,1,2,3,4,5,6,7,8)]
```