Disaster Relief Project (Part 2)_updated

Victor Teelucksingh - vat5jy - 8-7-2023

DS 6030 | Summer 2023 | University of Virginia Victor Teelucksingh - vat5jy - 8/7/2023

Section 1: Introduction

1.0: Project Description

The goal of this project is to use the classification methods covered in this course to solve a real historical data-mining problem: locating displaced persons living in makeshift shelters following the destruction of the earthquake in Haiti in 2010. We want to determine an optimal statistical method to as accurately as possible, and in as timely a manner as possible, locate as many of the displaced persons so that they can be provided food and water before their situations become unsurvivable.

The provided training data consists of 63,241 observations representing the RGB values for pixels produced from high-resolution, geo-referenced imagery data collected by aircraft by the Rochester Institute of Technology during the relief efforts. Each pixel has a classification of Blue Tarp, Rooftop, Soil, Various Non-Tarp, and Vegetation based on the combination of the intensity of the Red, Green, and Blue values.Displaced persons were using blue tarps to create makeshift shelters during the disaster, so blue tarps were identified as good indicators of where the displaced persons were. Given this context, we are most concerned with determining whether given RGB values can identify a potential blue tarp pixel or not (i.e., binary classification). We will train models to predict if a pixel/location indicates a blue tarp, which could assist the relief volunteers in finding the displaced persons. We will be working with logistic regression, LDA, QDA, KNN, and penalized logistic models to identify the best model to solve our problem.

As we build these models, we will need to choose tuning parameters and thresholds that best optimize our goals in this context. We will document the performance of these models using 10-fold cross-validation. We will likely focus on optimizing for sensitivity and minimizing the false negative rate (FNR) because we are dealing with human life. Although, we will need to make some consideration to minimize the false positive rate (FPR) because we want to efficiently use our critical and limited relief resources (mostly military/volunteer support). Additionally, we want to maximize F1 score, which is a useful metric when considering imbalanced classification (blue tarps are much more rare than non-blue tarp pixels). We discuss in more detail in the following sections.

Section 2: Initial Data Wrangling and EDA (Training and Holdout data)

2.0: Load libraries, load data, develop exploratory graphs

We call libraries assist in our analysis for EDA (tidyverse, plotly, ggplot2, gridExtra), model building (caret), and summarizing results (knitr, kableExtra, pROC, MLmetrics).

```
#>
#>        Blue Tarp          Rooftop             Soil Various Non-Tarp
#>         3.197293        15.659145        32.520042         7.501463
#>       Vegetation
#>        41.122057
```

We examine the proportions of each class in the training data. We will use this to build a bar chart.

The proportion table and corresponding bar chart above give us a better idea of the distribution of the classifications in the data. We can see that Vegetation and Soil are the most common classes and Blue Tarp is the least common class. If we consider a binary classification where we consider Blue Tarp (Yes/No), we will likely have imbalanced classes during our model fitting and analysis of results. We will need to take that into consideration when deciding which statistics we should rely on for determining model thresholds and tuning parameters.

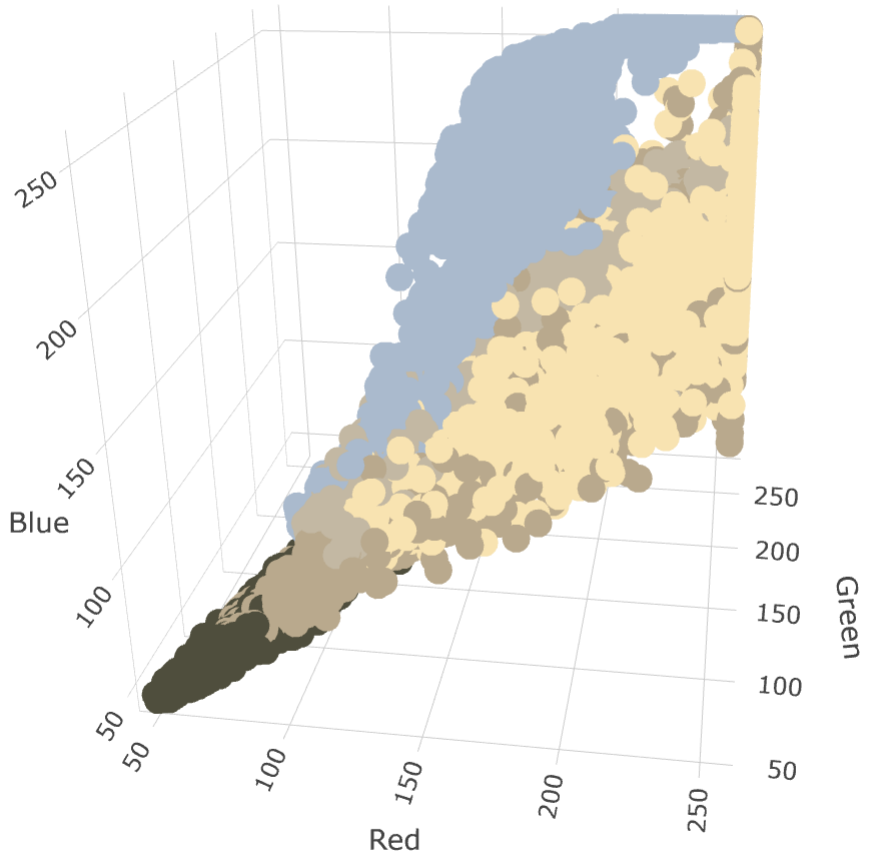
```
#>              Class  mean_red mean_green mean_blue
#> 1        Blue Tarp 169.66271  186.41494 205.03709
#> 2          Rooftop 195.42916  183.93810 162.80137
#> 3             Soil 247.86166  227.47306 176.70728
#> 4 Various Non-Tarp 184.82125  168.80164 140.92264
#> 5       Vegetation  78.99623   78.45497  60.92298
```

We find the mean RGB values across classes to better understand the distribution of RGB values.

Min/Max RGB Values in Training Data

color	min_vals	max_vals
Red	48	255
Green	48	255
Blue	44	255

We validated that our RGB values fall within the expected range (0 - 255) using min and max values for each RGB field.



I found the average intensity for the Red, Green, Blue values for each class. I converted the average RGB values into the corresponding hex color codes using an online converter to get better visual context on a colored representation for each class as it might appear on the geo-referenced imagery data. I used the hex codes to distinguish each class in a 3d plot of the observations to visualize the distribution of the classes. Although interesting, this scatterplot may be difficult to interpret, so we will create box plots below to more easily visualize the distribution of classes in the both the training and holdout data.

2.1: Load and clean holdout data. Compare with training data.

We load the data files to create the holdout set. I opened each file in a text editor to review the column names (if any) and delimiters, and to determine if any metadata rows should be skipped. 'orthovnir067_ROI_Blue_Tarps_data' is just the same dataset as 'orthovnir067_ROI_Blue_Tarps' but with fewer fields, so we can ignore this file.

We need to determine which fields align with our expected RGB values. Per the below tables, we examined the mean for each RGB column in the Blue Tarp data. Column 10 has the highest mean for each of the Blue Tarp datasets, so we will consider this the blue pixel field. We see that for the non blue tarp datasets that the first column has the highest mean across the board. When comparing to the mean of the RGB values for the non blue

tarp class in the training data in the table above, we see that the mean value for red is greater than the mean value for both blue and green, so we will consider column 8 as the red pixel field. By process of elimination we will assign column 9 as the green pixel field.

Column Means for Blue Tarp Data

col_headers	Blue_Tarps_067	Blue_Tarps_069	Blue_Tarps_078
V8	113.6628	122.6350	85.58671
V9	143.4748	138.3809	108.80287
V10	176.8180	168.5069	142.32252

Column Means for Non Blue Tarp Data

col_headers	NON_Blue_Tarps_057	NOT_Blue_Tarps_067	NOT_Blue_Tarps_069	NON_Blue_Tarps_078
V8	107.85187	122.40493	120.26914	138.8347
V9	90.01035	115.19505	113.72328	127.7839
V10	63.26049	92.22767	98.31425	106.2341

We add back in the column headers to the holdout data. We also create a binary variable to classify whether a pixel is a blue tarp pixel or not. We are primarily interested in training models to identify blue tarp pixels, and are less interested in examining the specifics of pixels that aren't blue tarp, so a binary variable works for our purposes. We will build this binary variable in both the training and holdout data sets.

```
#Add back in the column headers to each data set
colnames(ROI_057_NON_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Gree
n", "Blue")
colnames(ROI_067_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Green",
"Blue")
colnames(ROI_067_NOT_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Gree
n", "Blue")
colnames(ROI_069_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Green",
"Blue")
colnames(ROI_069_NOT_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Gree
n", "Blue")
colnames(ROI_078_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Green",
"Blue")
colnames(ROI_078_NON_Blue_Tarps) <- c("ID", "X","Y","Map X", "Map Y", "Lat", "Lon", "Red", "Gree
n", "Blue")
```

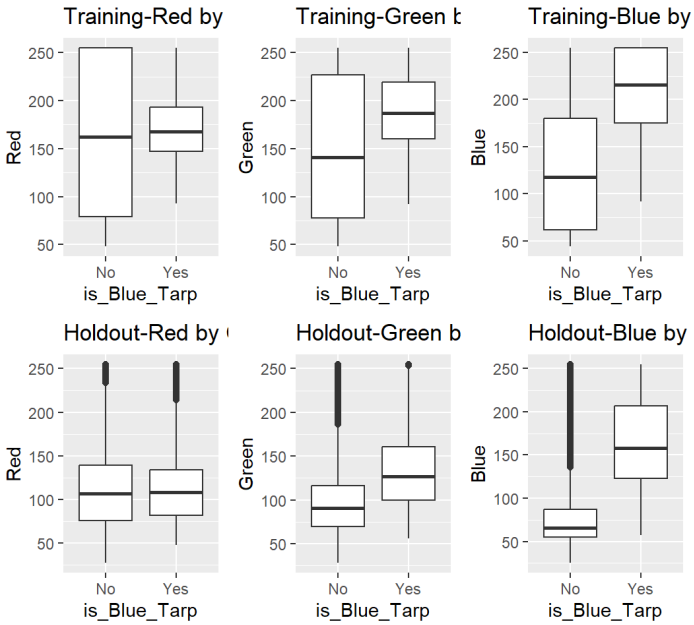
```
#Create binary variable (pixel classified as blue tarp or not)
Haiti$is_Blue_Tarp <- Haiti$Class
Haiti$is_Blue_Tarp[Haiti$is_Blue_Tarp!="Blue Tarp"] <- "No"
Haiti$is_Blue_Tarp[Haiti$is_Blue_Tarp=="Blue Tarp"] <- "Yes"

#Data format as factor
Haiti$is_Blue_Tarp <- as.factor(Haiti$is_Blue_Tarp)
```

```
#Create binary variable (pixel classified as blue tarp or not) for each dataset
ROI_057_NON_Blue_Tarps$is_Blue_Tarp <- "No"
ROI_057_NON_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_057_NON_Blue_Tarps$is_Blue_Tarp)

ROI_067_NOT_Blue_Tarps$is_Blue_Tarp <- "No"
ROI_067_NOT_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_067_NOT_Blue_Tarps$is_Blue_Tarp)

ROI_069_NOT_Blue_Tarps$is_Blue_Tarp <- "No"
ROI_069_NOT_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_069_NOT_Blue_Tarps$is_Blue_Tarp)

ROI_078_NON_Blue_Tarps$is_Blue_Tarp <- "No"
ROI_078_NON_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_078_NON_Blue_Tarps$is_Blue_Tarp)

ROI_067_Blue_Tarps$is_Blue_Tarp <- "Yes"
ROI_067_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_067_Blue_Tarps$is_Blue_Tarp)

ROI_069_Blue_Tarps$is_Blue_Tarp <- "Yes"
ROI_069_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_069_Blue_Tarps$is_Blue_Tarp)

ROI_078_Blue_Tarps$is_Blue_Tarp <- "Yes"
ROI_078_Blue_Tarps$is_Blue_Tarp <- as.factor(ROI_078_Blue_Tarps$is_Blue_Tarp)
```

Combine cleaned holdout data into one dataframe.

```
holdout_data <- rbind(ROI_057_NON_Blue_Tarps,ROI_067_NOT_Blue_Tarps,ROI_069_NOT_Blue_Tarps,ROI_0
78_NON_Blue_Tarps,
                      ROI_067_Blue_Tarps, ROI_069_Blue_Tarps, ROI_078_Blue_Tarps)
```

Before training and testing, we will perform EDA to compare the distributions in the training and holdout data.



We see some differences in the distributions for each color between the training and holdout data. The training data has significantly higher intensity than the holdout for Red based on median for both the positive and negative class. Likewise, the negative class in the training data has a wider distribution of Red than the holdout data, which has red clustered around the 75-140 range for both positive and negative classes.

We see a similar trend for both Green and Blue where the training data generally has higher intensity values for both the positive and negative classes when compared to the holdout data. The negative class also has a wider distribution of Green and Blue in the training data.

Generally both the training data and the holdout follow similar trends where the positive class has noticeably more blue and green than the negative class. However, the general intensity is higher for the training data across all colors than for the holdout data. The training data also has wider distributions of colors, particularly for the negative class. This may have an eventual impact during holdout testing because our models are trained to identify classes using higher intensities. I would expect that this would translate into an issue for our eventual test FNR because we may be failing to identify blue tarps in the test data simply because the color intensity is too low across the board.

Section 3: Model Fitting, Tuning Parameter Selection, and Evaluation

3.0: Set random seed & setup cross-validation folds

```
#Data wrangling and set-up
#References:

# Set random seed to allow for reproducible results.
# The CV folds are not created during defining trainControl unless explicitly stated using the i
ndex argument. We can use the createMultiFolds() function to create our CV folds for repeated us
e.

set.seed(1)

myfolds <- createMultiFolds(y = Haiti$is_Blue_Tarp, k=10, times=1)

# Specify type of training method used and the number of folds. In caret, we define the cross-va
lidation using the trainControl function. In this case, we specify CV and use the folds we creat
ed above as the index.
ctrlspecs <- trainControl(method="cv",
                          index=myfolds,
                          savePredictions=TRUE,
                          classProbs=TRUE,
                          summaryFunction = twoClassSummary)
```

We set a random seed to allow for reproducible results. We use create the CV folds for repeated use in each model. We used the createMultiFolds() function with 10 folds following generally accepted standards for cross-validation, and set times=1 to indicate no repetitions to limit calculation time in R. We then save our control specifications in the crtlspecs variable to be used in each model. [1] [2]

Each model family will follow the same formula:

$$\mathbf{is\_Blue\_Tarp} = RedX\_1 + GreenX\_2 + BlueX\_3$$

Other than creating the binary is_Blue_Tarp variable, there was no additional pre-processing (e.g., scaling) for any of the models. The RGB values already have a defined scaling from 0-255. Transformations and interactions may not fit well given the context, as the predictors are simply numeric values identifying the intensity of RGB colors for a pixel (even if model results may indicate better performance using transformations or interactions, it's difficult to justify using those transformations in this context.) Because we do not train our models with transformations or interactions, no further pre-processing was needed per in-class discussions.

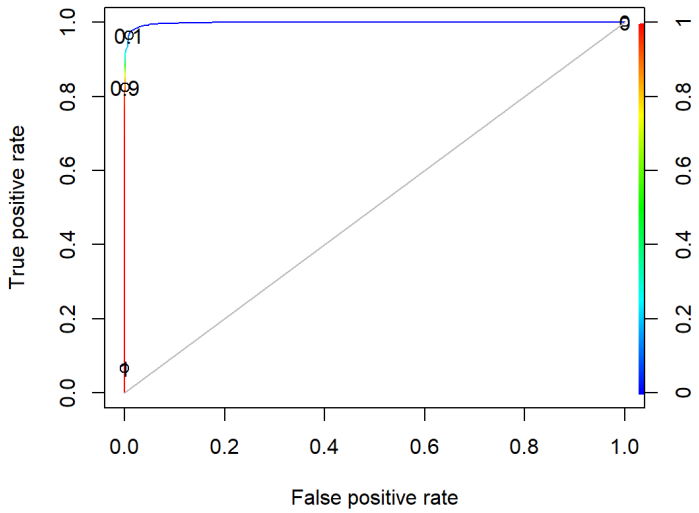
3.1 Fit Logistic Model

```
# Fit Logistic model
glm.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, family="binomial", method="glm",
trControl=ctrlspecs)

summary(glm.fit)
```

```
#>
#> Call:
#> NULL
#>
#> 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
```

Per EDA, we know that we have unequal distribution of classes in our training data, so we are dealing with "imbalanced classification." In this case, classification accuracy is a less reliable statistic because high accuracy can be achieved by a model that simply predicts the majority class. One strategy for choosing an appropriate threshold in an imbalanced classification scenario is to use metrics that focus on one class, like sensitivity or specificity. Given the context, sensitivity may be a better statistic for deciding the appropriate threshold because it is concerned with how well the positive/minority class (is_Blue_Tarp = "Yes") is predicted. We are more concerned with limiting the FNR than we are the FPR because we want ensure we are not leaving anyone behind given we are dealing with preventing loss of human life. Although a higher FPR potentially means a decrease in efficiency with additional resources spent, we can reliably assume that this project is an iterative step in finding the displaced persons (i.e., this is a starting point where additional research will be made to confirm people are stranded at chosen locations). While we will focus on optimizing sensitivity and FNR, we will also consider F-1 score which is useful in imbalanced classification because it balances statistics using both the minority and majority classes. Our approach for choosing model thresholds will be to optimize F-1 and sensitivity while making some considerations to limit FPR. [3] [4] [5]

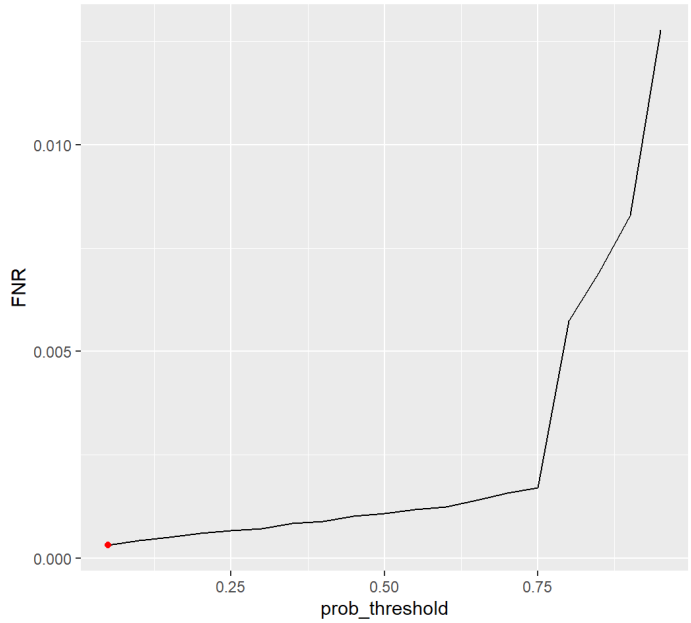
For this logistic model, we chose a threshold of .65 to maximize F1 while still maintaining high sensitivity / low FNR. We also chose this threshold to consider having a lower FPR to better utilize relief resources.




```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

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

Tradeoffs between TPR an FPR appear as expected in the above ROC curve. AUC = .9985069 which indicates high predictive performance across all thresholds.



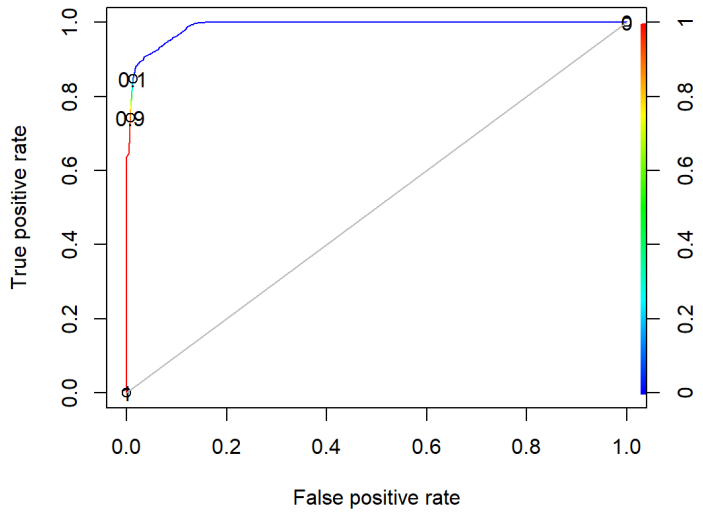
We see that FNR drastically increases for thresholds above .75, further justifying our choice of threshold = .65.

3.2 Fit LDA Model

```
#LDA (Linear Discriminant Analysis)
lda.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method="lda", trControl=ctrlspecs)

lda.fit
```

```
#> Linear Discriminant Analysis
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56916, 56917, 56917, 56916, 56917, 56917, ...
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.9888399  0.9899378  0.8011828
```

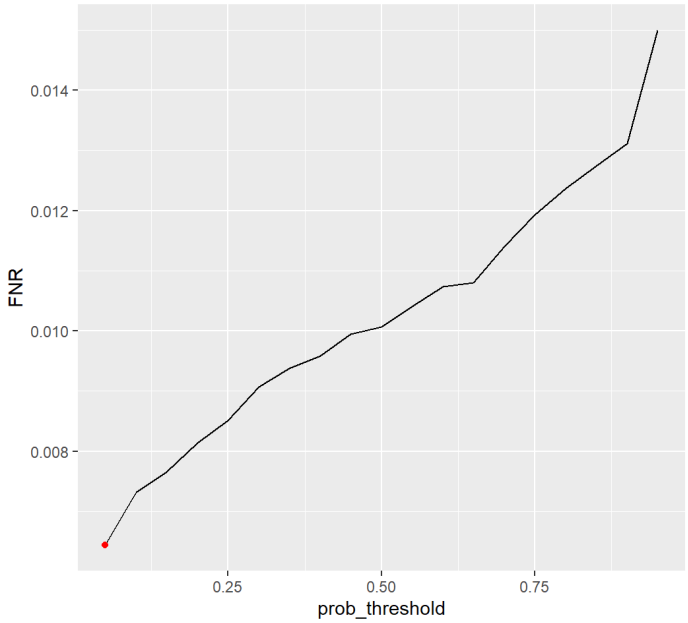
For this LDA model, we chose a threshold of .55 to maximize F1 while still maintaining high sensitivity / low FNR. We also chose this threshold to consider having a lower FPR to better utilize relief resources.




```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

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

We see a slight difference in the shape of the ROC curve for LDA compared to logistic indicating a slight loss in performance. AUC = .9888768 indicates high predictive performance across all thresholds, although this is slightly lower than logistic.



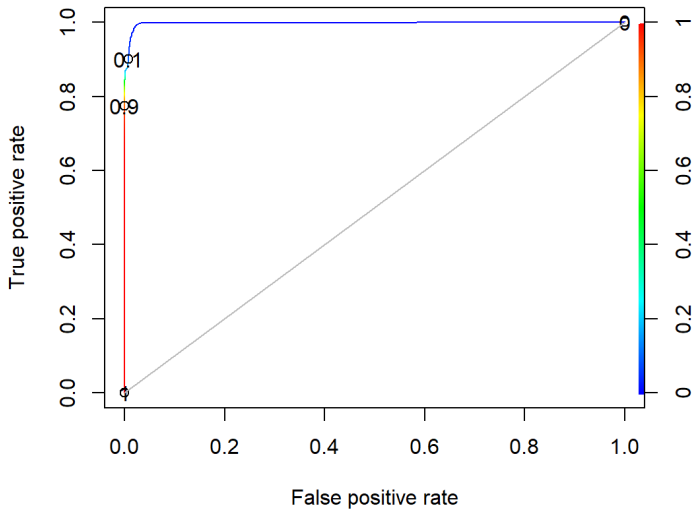
FNR steadily increases across thresholds with some larger spikes at thresholds above .60. We choose a balanced threshold of .55 to minimize FNR while also considering implications to limit FPR.

3.3 Fit QDA Model

```
#QDA (Quadratic Discriminant Analysis)
qda.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method="qda", trControl=ctrlspec
s)

qda.fit
```

```
#> Quadratic Discriminant Analysis
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56916, 56917, 56917, 56916, 56917, 56917, ...
#> Resampling results:
#>
#>   ROC        Sens      Spec
#>   0.9982039  0.999706  0.8402502
```

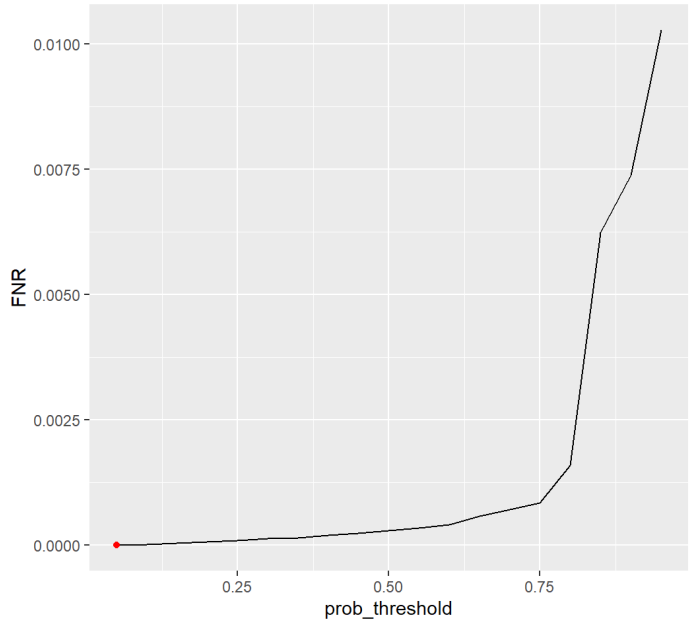
For this QDA model, we chose a threshold of .80 to maximize F1 while still maintaining high sensitivity / low FNR. We also chose this threshold to consider having a lower FPR to better utilize relief resources.




```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

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

The ROC curve shows improvement over LDA and closer to the shape of the ROC curve for logistic. AUC = .9982175 indicates high predictive performance across all thresholds.



FNR stays relatively constant but has a large increase at thresholds above .80. This further justifies our choose of a balanced threshold of .80 to minimize FNR while also considering implications to limit FPR.

3.4 Fit KNN Model

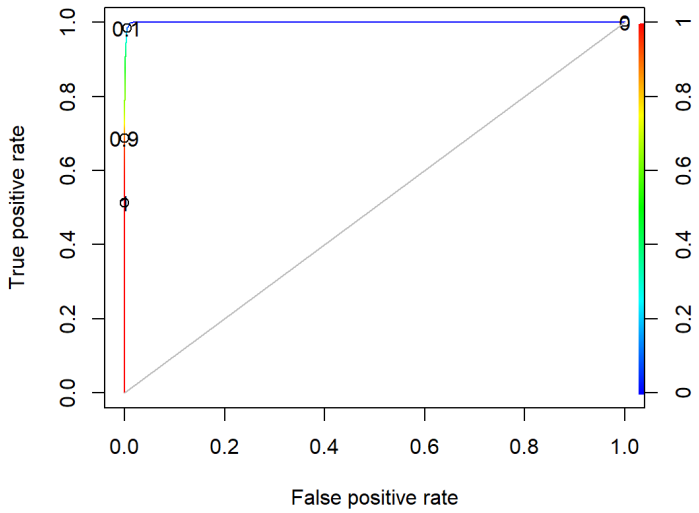
```
#KNN (K-nearest neighbor)
#Update with metric using sensitivity.

knn.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method="knn", trControl=ctrlspecs, metric="Sens", tuneGrid = expand.grid(k = c(5, 10, 20, 251)))
knn.fit
```

```
#> k-Nearest Neighbors
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56916, 56917, 56917, 56916, 56917, 56917, ...
#> Resampling results across tuning parameters:
#>
#>   k    ROC        Sens       Spec
#>     5  0.9983736  0.9984482  0.9584597
#>    10  0.9992037  0.9983828  0.9624128
#>    20  0.9995010  0.9983828  0.9564820
#>   251  0.9995077  0.9989382  0.8610179
#>
#> Sens was used to select the optimal model using the largest value.
#> The final value used for the model was k = 251.
```

```
#fit knn model using k = 5,10,20,251 as tuning parameters.
```

We optimized sensitivity to find k=251 as the ideal tuning parameter through cross-validation. We chose to consider k=5,10,20 which are typical values for k, as well as k=251=sqrt(n)=sqrt(63241) which is another standard choice for k in the field of data science. [6]

For this KNN model, we chose a threshold of .75 to maximize F1 while still maintaining high sensitivity / low FNR. We also chose this threshold to consider having a lower FPR to better utilize relief resources.

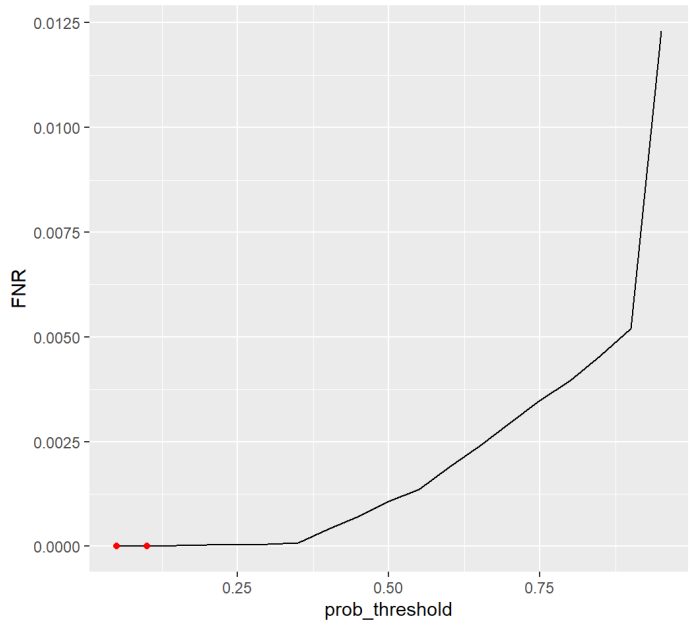



```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

```
#> [1] 0.9995206
```

The ROC curve shows improvements compared to logistic, LDA, and QDA. AUC = 0.9995206 indicates high predictive performance across all thresholds.

```
minFNR <- threshold.stats %>% slice_min(FNR)
ggplot(threshold.stats, aes(x=prob_threshold, y=FNR)) +
  geom_line() +
  geom_point(data=minFNR, color="red")
```



FNR steadily increases from 0 to ~.80 and then increases drastically. This further justifies our choose of a balanced threshold of .75 to maximize F1, minimize FNR, while also considering implications to limit FPR.

3.5 Fit Penalized Logistic Model

```
# Penalized Logistic Regression (elastic net penalty)

# Define the grid of tuning parameters to explore
# Choose Sensitivity as deciding metric for optimal alpha/lambda combination.

ctrlspecs_2 <- trainControl(method="cv",
                            index=myfolds,
                            savePredictions=TRUE,
                            classProbs=TRUE,
                            summaryFunction = prSummary)
lambdas=10^seq(-4,2,0.4)
alphas = seq(0,1,by=0.1)
tuneGrid <- expand.grid(alpha=alphas,
                        lambda=lambdas)

modelFit <- train(is_Blue_Tarp ~ Red + Green + Blue,
                  data=Haiti,
                  method='glmnet',
                  metric="F",
                  trControl=ctrlspecs_2,
                  tuneGrid=tuneGrid)
```

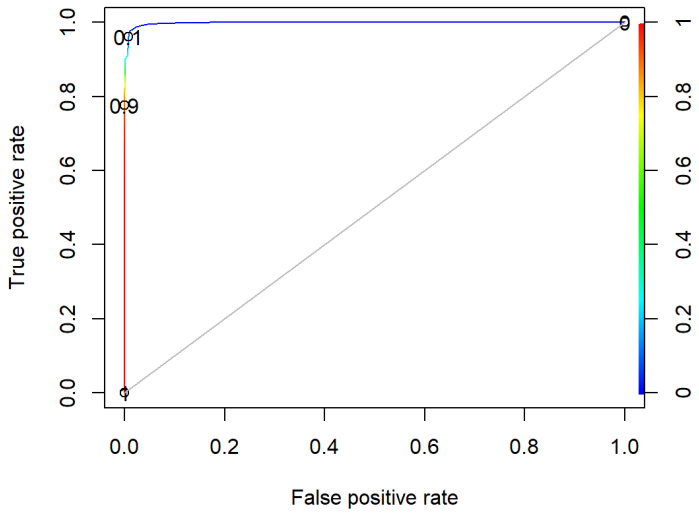
Sensitivity does not seem to vary drastically in this model for given alpha/lambda values (usually close to 1.0). We choose our ideal alpha/lambda values by maximizing F1, which seems to vary much depending on chosen alpha/lambda. I also ran into some performance issues for this model optimizing for Sensitivity, where my model reported Sensitivity 1 across all thresholds. So we will maximize F1 score while also making considerations for Sensitivity and FPR

```
modelFit$results %>% slice_max(F)
```

```
#>   alpha lambda       AUC Precision    Recall         F        AUCSD
#> 1     1  1e-04 0.9997361 0.9956548 0.9992976 0.9974728 7.976782e-05
#>    PrecisionSD     RecallSD          FSD
#> 1 0.0005719848 0.0002781731 0.0003117732
```

For this penalized logistic model, we chose a threshold of .70 to maximize F1 while still maintaining high sensitivity / low FNR. We also chose this threshold to consider having a lower FPR to better utilize relief resources.

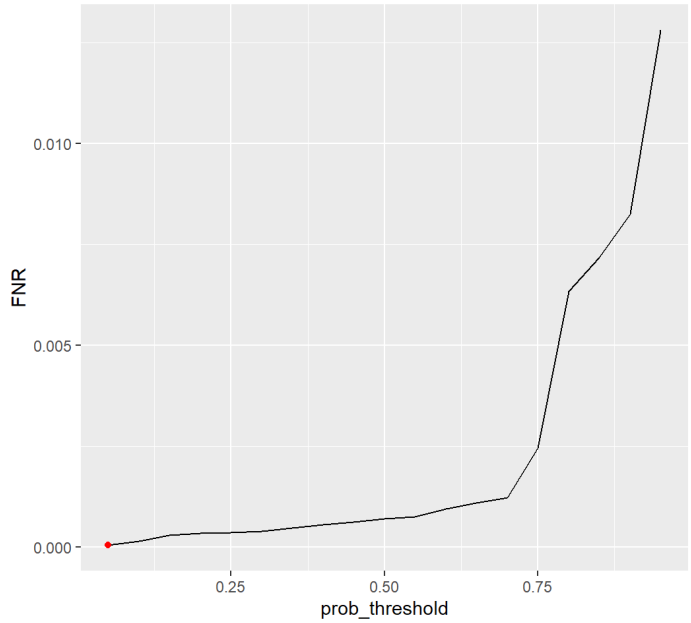
We choose tuning parameters of alpha = 1 and lambda = 1e-04 to maximize F1.




```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

```
#> [1] 0.9984959
```

ROC curve is similar to logistic, LDA, QDA, but not as highly performing as KNN. AUC = 0.9984959 indicates high predictive performance across all thresholds.



FNR stays relatively constant from 0 to ~.75 and then increases drastically. This further justifies our choose of a balanced threshold of .70 to minimize FNR while also considering implications to limit FPR.

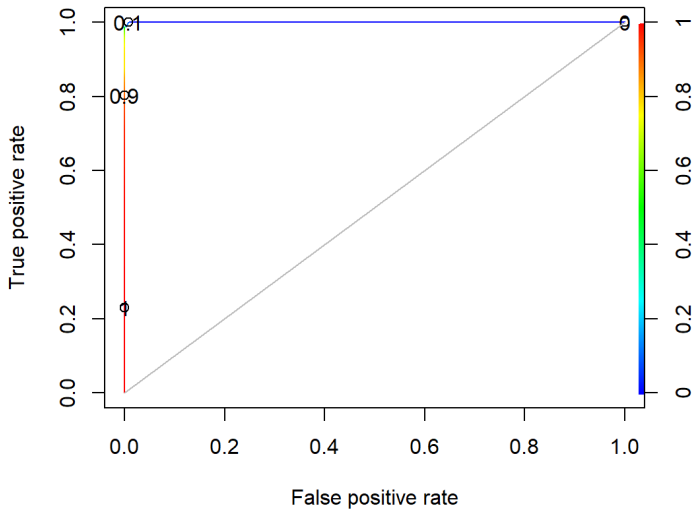
3.6 Random Forest Model

```
#train RF + SVM models
set.seed(55)


#tune with 1,sqrt(p=3) => 2, and p=3 for mtry
rf_tuneGrid <- data.frame(mtry= c(1,2,3))

rf.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method='rf',
                        metric="Sens",
                        tuneGrid = rf_tuneGrid,
                        trControl=ctrlspecs)


svm.poly.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method='svmPoly',
                        metric="Sens",
                        trControl=ctrlspecs)

svm.linear.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method='svmLinear',
                        metric="Sens",
                        trControl=ctrlspecs)

svm.radial.fit <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method='svmRadial',
                        metric="Sens",
                        trControl=ctrlspecs)
```

For the Random Forest model, we chose a threshold of .50 to maximize F1 while still maintaining high sensitivity / low FNR. The FPR still remains relatively low at this threshold. We considered mtry = 1, mtry = sqrt(p=3) => 2, and mtry = p = 3 per textbook recommendations for optimizing mtry for a classification model. The model chose mtry = 1 as our ideal tuning parameter to optimize Sensitivity based on 10-fold cross validation.

```
threshold.stats %>% slice_max(F1)
```

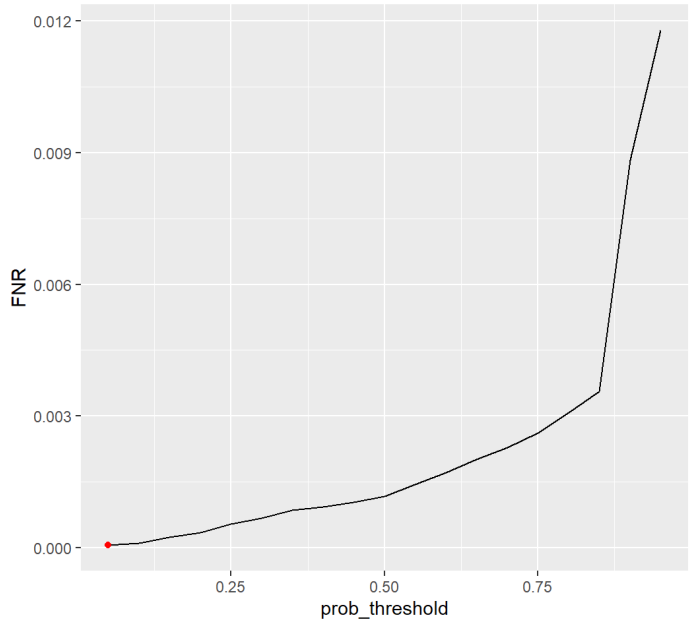
```
#>   mtry prob_threshold Sensitivity Specificity Pos Pred Value Neg Pred Value
#> 1    1            0.5   0.9988402   0.9470858      0.9982534      0.9644198
#>   Precision    Recall        F1 Prevalence Detection Rate Detection Prevalence
#> 1 0.9982534 0.9988402 0.9985466  0.9680271      0.9669044            0.9685963
#>   Balanced Accuracy  Accuracy     Kappa        J       Dist         FNR
#> 1          0.972963 0.9971854 0.9541335 0.945926 0.05292914 0.001159784
#>          FPR
#> 1 0.05291421
```




```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

```
#> [1] 0.9999451
```

The ROC curve for the Random Forest model shows an improvement over logistic, LDA, QDA, KNN, and penalized logistic. AUC = 0.9999451 indicates high predictive performance across all thresholds.



We can see that FNR spikes past .75. Using a prob_threshold at .50 is a compromise for a low FNR, maximizing F1 score, while also maintaining a low FPR.

3.7 Support Vector Machines (SVM)

We trained SVM models using linear, polynomial, and radial kernels using default tuning to optimize F1 and Sensitivity. We will choose the model that has the highest F1 score and continue with additional tuning.

Linear:

```
threshold.stats %>% slice_max(F1)
```

```
#>   C prob_threshold Sensitivity Specificity Pos Pred Value Neg Pred Value
#> 1 1            0.8   0.9980888   0.9248281      0.9975188      0.9414118
#>   Precision    Recall        F1 Prevalence Detection Rate Detection Prevalence
#> 1 0.9975188 0.9980888 0.9978036  0.9680271       0.966177            0.9685805
#>   Balanced Accuracy  Accuracy     Kappa         J       Dist         FNR
#> 1         0.9614585 0.9957465 0.9307243 0.9229169 0.07519918 0.001911164
#>          FPR
#> 1 0.07517193
```

Polynomial:

```
threshold.stats %>% slice_max(F1)
```

```
#>   degree scale    C prob_threshold Sensitivity Specificity Pos Pred Value
#> 1      1   0.1 0.25           0.55   0.9992813   0.8664708       0.995606
#>   Neg Pred Value Precision    Recall        F1 Prevalence Detection Rate
#> 1      0.9755723  0.995606 0.9992813 0.9974402  0.9680271      0.9673313
#>   Detection Prevalence Balanced Accuracy  Accuracy     Kappa        J      Dist
#> 1            0.9716007          0.932876 0.9950349 0.9151705 0.865752 0.1335315
#>            FNR       FPR
#> 1 0.0007187354 0.1335292
```

Radial:

```
threshold.stats %>% slice_max(F1)
```

```
#>      sigma   C prob_threshold Sensitivity Specificity Pos Pred Value
#> 1 8.733912 0.5            0.9   0.9982195   0.9604375      0.9986927
#>   Neg Pred Value Precision    Recall       F1 Prevalence Detection Rate
#> 1      0.9471019 0.9986927 0.9982195 0.998456  0.9680271      0.9663035
#>   Detection Prevalence Balanced Accuracy  Accuracy     Kappa        J
#> 1            0.9675685         0.9793285 0.9970114 0.9520877 0.958657
#>         Dist         FNR       FPR
#> 1 0.03960915 0.001780491 0.0395625
```

The radial SVM model had the highest F1 score at .998, compared to .9978 using the linear kernel and .9974 for the polynomial kernel. For the polynomial kernel, maximizing F1 comes with the disadvantage of a relatively high FPR at .136. The radial model has a higher F1, higher Sensitivity, and lower FPR than the linear model, so we will go with the radial model. Default model tuning used sigma = 8.733912 and C = 0.5.

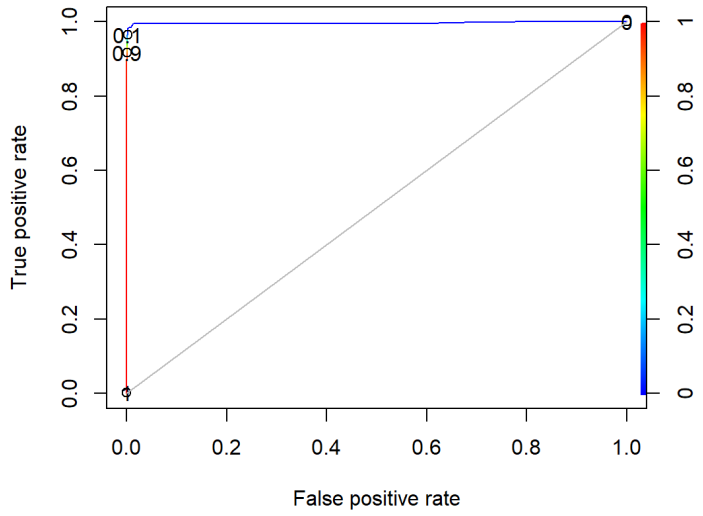
We will use cross-validation to consider other tuning metrics for sigma and C in the radial model.

```
set.seed(7)
svm.radial.fit_tuned <- train(is_Blue_Tarp ~ Red + Green + Blue, data=Haiti, method='svmRadial',
                        metric="Sens",
                        tuneGrid=expand.grid(C=c(.5, 1, 1.5, 2, 2.5, 3), sigma=c(1,5,7,8,9,10)),
                        trControl=ctrlspecs)
```

```
#>   sigma   C prob_threshold Sensitivity Specificity Pos Pred Value
#> 1     9 1.5           0.85   0.9984972   0.9594498      0.9986604
#>   Neg Pred Value Precision    Recall        F1 Prevalence Detection Rate
#> 1      0.9549075 0.9986604 0.9984972 0.9985787  0.9680271      0.9665723
#>   Detection Prevalence Balanced Accuracy  Accuracy     Kappa        J
#> 1             0.967869         0.9789735 0.9972486 0.9556868 0.957947
#>         Dist         FNR        FPR
#> 1 0.04058257 0.001502801 0.04055016
```

In the tuned radial model, we see that the chosen tuning is sigma = 9 and C = 1.5. We considered values for C in .5 increments from .5 to 3, as well as sigma = 1,5,7,8,9,10. We chose these values for C because increasing C only improves performance to a limit. We also chose various sigma values close to the default to see how sigma would impact performance. F-1 score and Sensitivity marginally improve at the expense of a marginal loss of performance in FPR. We will go with this tuned model that optimizes F-1 score and Sensitivity at the expense of FPR.

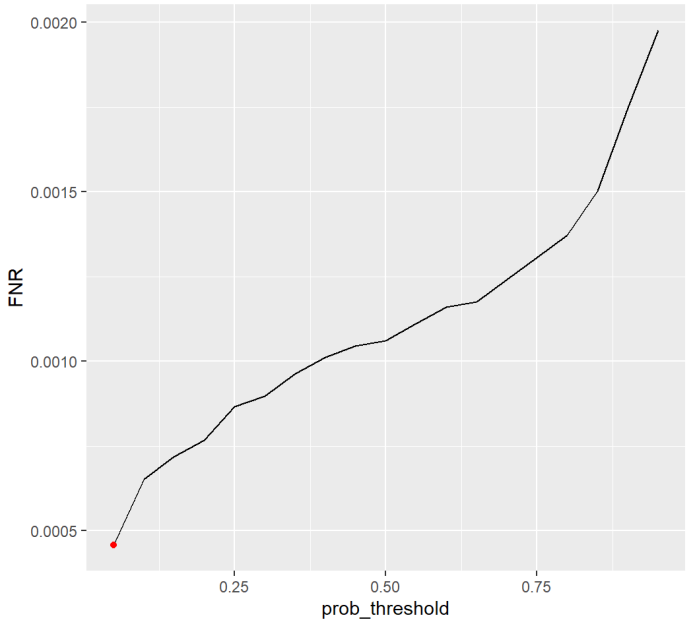
We choose a threshold of .80 which maximizes F-1 score at 0.999 while maintaining a high Sensitivity at 0.999 with considerations for a lower FPR at 0.045.




```
AUC <- ROCR::performance(predob,"auc")@y.values[[1]]
AUC
```

```
#> [1] 0.9964509
```

The ROC curve indicates a slight decrease in performance compared to the RF model. AUC = 0.9964509 indicates high predictive performance across all thresholds.



FNR steadily increases as the probability threshold increases. The fitted line gets steeper after our selected threshold of .80. However, there is not a drastic difference in magnitude between FNR at higher thresholds compared to lower thresholds. We maximize F-1 score and Sensitivity with some considerations for maintaining a relatively low FPR.

Section 4

4.0 Cross-Validation Performance Table

```
Performance_Table <- read.csv('updated_training_perf_.csv')
knitr::kable(Performance_Table,"pipe")
```

Model	Tuning.Parameters	AUC	Selected.threshold	F1	Accuracy	TPR	FPR	Precision
Logistic	NA	0.9985100	0.65	0.998	0.996	0.999	0.098	0.997
LDA	NA	0.9888800	0.55	0.992	0.984	0.990	0.194	0.994
QDA	NA	0.9982200	0.80	0.997	0.994	0.998	0.132	0.996
KNN	k=251	0.9995206	0.75	0.998	0.995	0.997	0.044	0.999
Penalized Logistic	alpha = 1, lambda = 1e-04	0.9984959	0.70	0.998	0.995	0.999	0.106	0.997
Random Forest	mtry = 1	0.9999451	0.50	0.999	0.997	0.999	0.053	0.998
SVM (radial kernel)	sigma = 9, C = 1.5	0.9964509	0.80	0.999	0.997	0.999	0.045	0.998

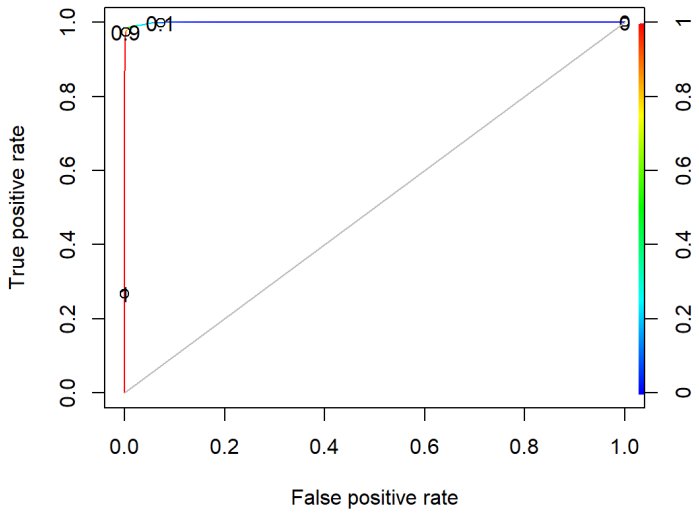
See above for table of results developed through cross-validation. These results will be discussed in the Results and Conclusions section.

Section 5: Test Holdout set

See below for process used to test the holdout set with each trained model above. ROC curves for the holdout set will be plotted below and statistics will be summarized in the resulting performance table.

5.0: Logistic

```
predob <- ROCR::prediction(holdout_data$prob, holdout_data$is_Blue_Tarp)
model.roc <- ROCR::performance(predob, measure='tpr', x.measure='fpr')
plot(model.roc, downsampling=0.001, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0))
lines(x=c(0,1), y=c(0,1), col='grey')
```



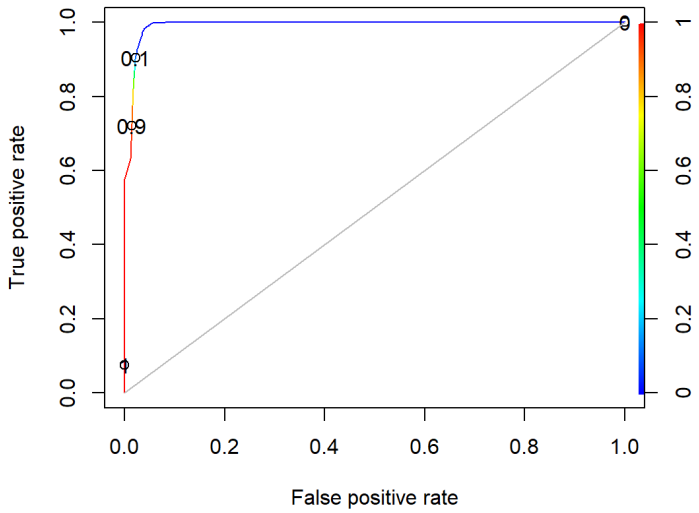

```
glm.auc
```

```
#> Area under the curve: 0.9994
```

The ROC curve is shaped well and AUC = 0.9994 indicating strong performance.

5.1: LDA

```
predob <- ROCR::prediction(holdout_data$prob, holdout_data$is_Blue_Tarp)
model.roc <- ROCR::performance(predob, measure='tpr', x.measure='fpr')
plot(model.roc, downsampling=0.001, colorize=T, print.cutoffs.at=c(0, 0.1, 0.9, 1.0))
lines(x=c(0,1), y=c(0,1), col='grey')
```

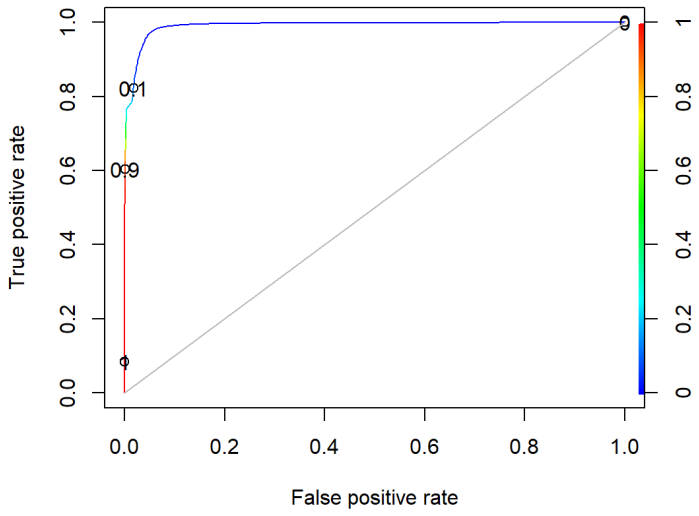



```
lda.auc
```

```
#> Area under the curve: 0.9921
```

We can see above that there is a dip in the ROC curve indicating a drop in performance on the holdout set using LDA vs. logistic regression. AUC = .9921 which is slightly lower than logistic.

5.2: QDA

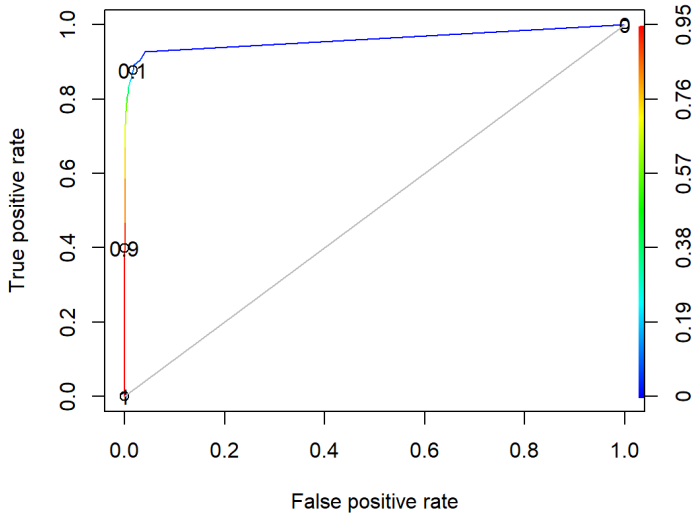



```
qda.auc
```

```
#> Area under the curve: 0.9915
```

The QDA ROC curve shows a slight drop in performance compared to both LDA and logistic. AUC decreases slightly to .9915, but this is still indicative of strong performance.

5.3: KNN

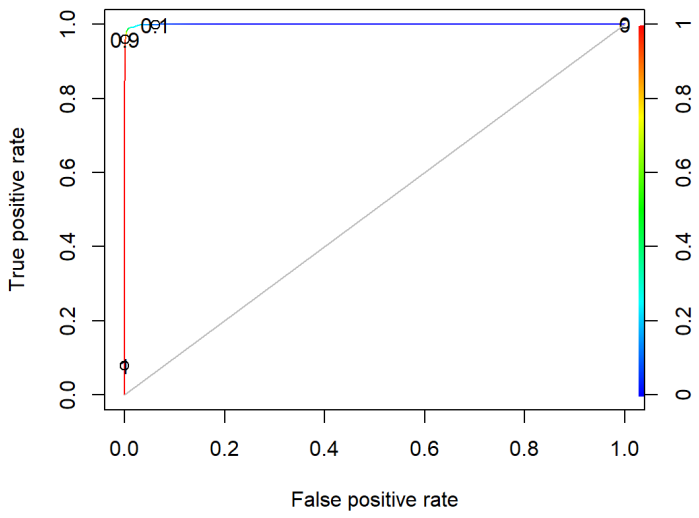



```
knn.auc
```

```
#> Area under the curve: 0.9609
```

Per the above ROC curve, we see that the KNN model performed slightly worse than the previous models on the holdout set. AUC = .9609 which indicates a potential loss in performance.

5.4: Penalized Logistic

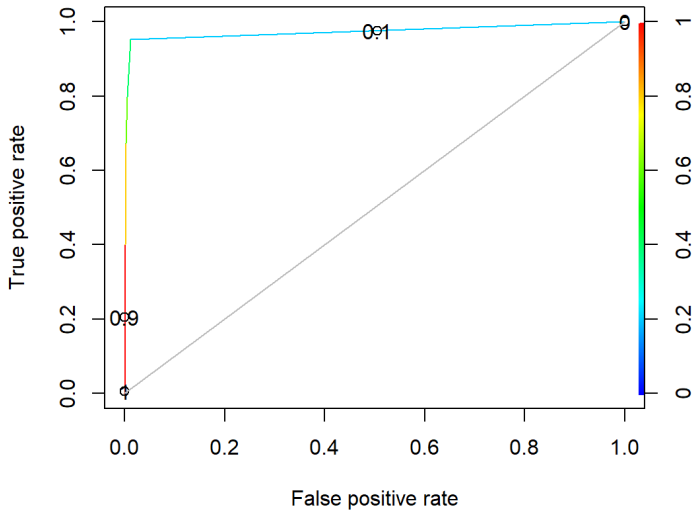



```
penalized_logistic.auc
```

```
#> Area under the curve: 0.9996
```

The above ROC curve shows an increase in performance compared to all prior models. We see that AUC = .9996 which indicates strong predictive performance of penalized logistic on the holdout set.

5.5: Random Forest

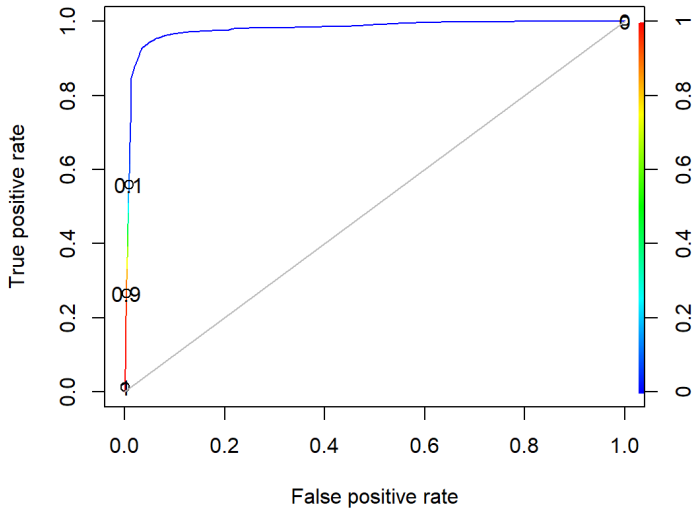



```
rf.auc
```

```
#> Area under the curve: 0.9823
```

The above ROC curve and AUC = .9823 suggest a drop in performance on the holdout set compared to penalized logistic results.

5.6: Support Vector Machine (SVM) - Radial Kernel




```
svm.auc
```

```
#> Area under the curve: 0.979
```

Per the above ROC curve and AUC, we see a decrease in performance compared to other models. However, this model may outperform KNN.

Section 6: Results

Cross-validated training results:

```
Performance_Table <- read.csv('updated_training_perf_.csv')
knitr::kable(Performance_Table,format="pipe")
```

Model	Tuning.Parameters	AUC	Selected.threshold	F1	Accuracy	TPR	FPR	Precision
Logistic	NA	0.9985100	0.65	0.998	0.996	0.999	0.098	0.997
LDA	NA	0.9888800	0.55	0.992	0.984	0.990	0.194	0.994
QDA	NA	0.9982200	0.80	0.997	0.994	0.998	0.132	0.996
KNN	k=251	0.9995206	0.75	0.998	0.995	0.997	0.044	0.999
Penalized Logistic	alpha = 1, lambda = 1e-04	0.9984959	0.70	0.998	0.995	0.999	0.106	0.997
Random Forest	mtry = 1	0.9999451	0.50	0.999	0.997	0.999	0.053	0.998
SVM (radial kernel)	sigma = 9, C = 1.5	0.9964509	0.80	0.999	0.997	0.999	0.045	0.998

Holdout set performance results:

Model	Tuning.Parameters	AUC	Selected.threshold	F1	Accuracy	TPR	FPR	Precision
Logistic	NA	0.9994	0.65	0.701	0.701	0.985	0.0060	0.544
LDA	NA	0.9921	0.55	0.398	0.982	0.832	0.0170	0.261
QDA	NA	0.9915	0.80	0.711	0.711	0.637	0.0010	0.804
KNN	k=251	0.9609	0.75	0.701	0.996	0.582	0.0005	0.882
Penalized Logistic	alpha = 1, lambda = 1e-04	0.9996	0.70	0.836	0.997	0.977	0.0030	0.730
Random Forest	mtry = 1	0.9823	0.50	0.680	0.995	0.775	0.0040	0.605
SVM (radial kernel)	sigma = 9, C = 1.5	0.9790	0.80	0.326	0.991	0.314	0.0040	0.339

Section 7: Conclusions

7.1.0: Determination and justification of which algorithm works best (cross-validation/training data)

Per the above, we created a tables using the optimal threshold and tuning parameters (if applicable) to best predict Blue Tarp status. In general, we chose parameters/thresholds to maximize F1, a balanced statistic, as well as maximizing sensitivity / minimizing FNR to ensure we are fully capturing possible blue tarp locations.

We first review the cross-validated training results. The Random Forest model has the highest AUC at .9999 and LDA has the lowest at .9888, although every model trained generally has high AUC. In order to maximize F1 and Sensitivity, we considered a large range of thresholds across the models, with SVM and QDA having high thresholds at .80 and Random Forest having the lowest chosen threshold at .50. In general, lower thresholds are more lenient in classifying RGB values as Blue Tarp pixels, which results in a higher sensitivity at the cost of a higher false positive rate. Our overall goal was to maximize F1 and Sensitivity, so I didn't always choose the lowest thresholds because we made some considerations to limit the FPR. This makes sense in the disaster relief scenario because we want to capture as many positive cases as possible while remaining realistic.

In the training results, we see that the Random Forest model and SVM model using the radial kernel have the highest F-1 scores across all results, as well as some of the lowest FPR. Random Forest and SVM are also on the high end for TPR/Specificity at .999 while maintaining high precision at .998 and accuracy at .997. In choosing between these two, I go with the Random Forest model because it has a higher AUC at the expense of a slightly higher FPR than the SVM model. This should improve model performance at the expense of some relief resources spent on false positives. So we maintain strong predictive performance using Random Forest and still maintain relatively low FPR compared to the other models.

I am not surprised that Random Forest yields the best cross-validation results compared to the other models. We learned in Module 11 that random forest classifiers are most likely to be high performs (at least when testing for Accuracy) compared to many other common algorithms. Likewise, based on our exploratory data analysis, the true relationship between the predictors and the response is likely different than the assumed relationships using Logistic, LDA, and QDA. We also know from class that Random Forest / bootstrapping should work well with a significant amount of training data (i.e., 63,241 observations).

7.1.1: Determination and justification of which algorithm works best (holdout data)

There is more noticeable variation between the result statistics in the holdout data results compared to the training data results. F1 score, Accuracy, and Precision seem to vary drastically depending on the chosen model. AUC still remains high across the board, although slightly lower than AUC observed during training. FPR actually improved using the holdout data compared to the training statistics.

We will continue to use the same logic of choosing the preferred model based on maximizing F1 score and Sensitivity. Using this logic, the holdout data results have a more clear winner: we choose the Penalized Logistic model (alpha = 1, lambda = 1e04) which has by far the highest F1 score at .836 as well as one of the highest Sensitivity at .977 (second only to logistic which has a Sensitivity of .985). The Penalized Logistic model also low FPR (only outperformed slightly by KNN and QDA), which will be useful in capturing as many positive cases as possible with a relatively low false positive rate to save critical relief sources.

Some models performed more poorly than expected on the holdout data. For example, LDA and SVM using the radial kernel had low F1 scores and precision (.398, .326; .261, .339, respectively). LDA is making a larger amount of false positive predictions with the highest FPR of all models at .0170. In contrast, SVM seems to be making a larger amount of false negative predictions with the lowest Sensitivity of all models at .314. These reasons drive the loss in F1 score for both LDA and SVM, although Accuracy remains high for both by predicting the majority class correctly.

7.2: Discussion justifying why findings are compatible or reconcilable.

We see some stark differences in results for the holdout set compared to results for the training set, particularly in Precision, TPR and F1, which is concerning because we trained our models and chose thresholds to optimize F1 and TPR. However, our findings are generally compatible/reconcilable because AUC remains very high performing for both the training results and the holdout results. AUC is a threshold invariant metric, meaning our choice of threshold does not change AUC. AUC is also a reliable metric during class imbalance situations (discussed more below), so it is a good sign that our AUC is high performing for all models in both the training and the holdout sets. [7]

The differences in training and holdout results may be explained in part by the differences in the distributions of the training and test data. During EDA, we saw that the training data uses higher intensity RGB values than our test data, which may have lead to some loss in performance in the holdout test results. We may see more compatible/reconcilable results if we use training/test sets that have more similar distributions.

7.3: Recommendation/rationale regarding optimal algorithm to use for detection of blue tarps

As discussed above, I ultimately would recommend using the penalized logistic regression algorithm in this context. Penalized logistic regression seemed to perform quite well across all metrics for both our training and test results. In particular, the penalized logistic regression model had the highest AUC and F1 score in the holdout results with the second highest TPR (only slightly below typical logistic regression) while maintaining a low FPR in both the training and test results. This shows high compatability between the training and holdout results, which gives me more confidence in using this algorithm for detection of blue tarps. Moreover, the high AUC for this model in both the training and test results makes it clear to me that it will be useful regardless of our choice of thresholding.

7.4: Discussion of the relevance of the metrics calculated above to this application context

In training, thresholding, and choosing an optimal model, we optimized for both F1 score and Sensitivity. F1 score is a useful middle-of-the-ground metric to assess model performance in an imbalanced classification context. Likewise, we want to also prioritize TPR/Sensitivity because we want to ensure we side more on identifying possible blue tarp pixels to keep the FNR low to save more lives. While this comes at the expense of a potentially higher FPR (i.e., wasting relief resources), this model gives a good balance between the two to ensure efficient use of resources.

Other metrics considered include Accuracy and Precision. Accuracy is less useful in this application context because simple models that predict just the majority class will likely have high Accuracy because blue tarp pixels are quite rare in comparison to the non blue tarp pixels (the majority class). Precision is a measure of the accuracy of positive predictions, which is also less relevant in this context because we less concerned with minimizing false positives (we would rather be more lenient in identifying blue tarps so we ensure we are not missing any displaced persons).

7.5: Challenges with Imbalanced Classification

Although we have touched on the issue of imbalanced classification in earlier sections, we will add some additional detail on conclusions here. Because Blue Tarp is identified in only ~3% of total training observations, we have a large skew in the data between the majority (“No” is not Blue Tarp) and minority (“Yes” is Blue Tarp) classes. This makes threshold and model selection more difficult as some commonly used statistics (i.e., Accuracy) become less reliable. As discussed in the additional resources on Canvas, the imbalanced classification issue further stresses the importance of sensitivity and specificity in the context of this project. Because we are primarily concerned with

limiting the FNR to avoid missing potentially displaced persons, we focus more on sensitivity than specificity. Given this context, misclassifying an observation from the majority class (false positive) is less critical than misclassifying an observation from the minority class (false negative). [8]

There are additional techniques we might use to improve results like random resampling to create a more balanced class distribution for our training models. We can use random oversampling (randomly create duplicates observations of the minority class) or random undersampling (randomly remove observations in the majority class) or a combination of the two to address the severe skew in the class distribution. This may lead to models with better predictive power and eventual improvements in holdout test results. [9]

7.6: Limitations of using RGB data for classification

RGB data are “sensitive to illuminations and shadows” and “cannot provide accurate representations of the shape of objects.” This may have skewed our classifications and may have had eventual ramifications on the predictive performance on the holdout data. Improvements to the data collection process may make use of more advanced 3D object representation approaches which may yield better performance for predictive modeling (i.e., RGB-D including depth as a feature). [10]

Using solely RGB values, we fail to capture potentially critical spatial information like depth that may improve classification accuracy and overall performance. Compared to RGB images, depth images are “robust to the variations in color, illumination, rotation angle and scale” which may have skewed training in our models. We are not given any detail on any of the image preprocessing techniques used by the researchers who developed the holdout data which may have had an averse effect on our models’ predictive power during testing. The provided example images from which the holdout data was derived show potential differences in altitude and shadowing from the local environment that are not fully considered when we only train models using RGB/color as predictors. Moreover, creating transformations or additional features using our raw RGB data is not intuitive unless you have some additional insight or training into image processing. In the context of our project, transformations or feature engineering using the existing RGB data likely lack contextual justification other than this appeared to improve model performance after guess and check. [11]

Section 8: Resources & References

1. https://stackoverflow.com/questions/52622811/does-using-the-same-traincontrol-object-for-cross-validation-when-training-multi (https://stackoverflow.com/questions/52622811/does-using-the-same-traincontrol-object-for-cross-validation-when-training-multi)↩
2. https://www.r-bloggers.com/2020/11/caretcreatefolds-vs-createmultifolds/ (https://www.r-bloggers.com/2020/11/caretcreatefolds-vs-createmultifolds/)↩
3. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/ (https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/)↩
4. https://towardsdatascience.com/optimal-threshold-for-imbalanced-classification-5884e870c293 (https://towardsdatascience.com/optimal-threshold-for-imbalanced-classification-5884e870c293)↩
5. https://neptune.ai/blog/balanced-accuracy (https://neptune.ai/blog/balanced-accuracy)↩
6. https://rpubs.com/Mentors_Ubiqum/tunegrid_tunelength (https://rpubs.com/Mentors_Ubiqum/tunegrid_tunelength)↩

7. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc# (https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#):~:text=AUC%20is%20scale%2Dinvariant.,what%20classification%20threshold%20is%20chosen.↩

8. https://machinelearningmastery.com/imbalanced-classification-is-hard/ (https://machinelearningmastery.com/imbalanced-classification-is-hard/)↩

9. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/ (https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/)↩

10. https://link.springer.com/article/10.1007/s11370-021-00349-8# (https://link.springer.com/article/10.1007/s11370-021-00349-8#):~:text=Object%20representations%20based%20on%20just,of%20the%20shape%20of%20objects.↩

11. https://www.sciencedirect.com/science/article/abs/pii/S0020025517300191 (https://www.sciencedirect.com/science/article/abs/pii/S0020025517300191)↩