Disaster_Recovery_Project_Part1

Jay Hombal

September 28, 2020

1. Introduction:

This project is a classification data-mining problem for locating displaced persons living in makeshift shelters following the destruction of the earthquake in Haiti in 2010.

Following that earthquake, rescue workers, mostly from the United States military, needed to get food and water to the displaced persons. But with destroyed communications, impassable roads, and thousands of square miles, actually locating the people who needed help was challenging.

As part of the rescue effort, a team from the Rochester Institute of Technology flew an aircraft to collect high-resolution geo-referenced imagery. It was known that the people whose homes had been destroyed by the earthquake were creating temporary shelters using blue tarps, and these blue tarps would be good indicators of where the displaced persons were - if only they could be located in time, out of the thousands of images that would be collected every day. The problem was that there was no way for aid workers to search the thousands of images in time to find the blue tarps and communicate the locations back to the rescue workers on the ground in time. The solution would be provided by data-mining algorithms, which could search the images faster and more thoroughly (and accurately?) then humanly possible.

The goal was to find an algorithm that could effectively search the images to locate displaced persons and communicate those locations rescue workers so they could help those who needed it in time.

2. Prepare Problem

a) Load packages

```
# load all required libraries
library(ISLR)
library(tidyverse)
library(yardstick)
library(caret)
library(recipes)
library(MASS)
library(pROC)
library(doParallel)
library(tune)
```

a) Optimize compute settings

```
# code shared by Derek - to improve speed
#https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf
cores <- parallel::detectCores()
cores
```

```
## [1] 12
```

```
all_cores <- parallel::detectCores(logical = FALSE)
all_cores
```

```
## [1] 6
```

```
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)
```

```
grid_control <- control_grid(verbose = TRUE,pkgs = "doParallel",allow_par = TRUE)
```

b) Intialize constants

```
# seed
seed = 0424

# define the filename
input_file  <- "data/HaitiPixels.csv"
```

c) Load haiti_ds

```
# load the CSV file fril the local directory
haiti_ds <- read.csv(input_file, header= TRUE, sep=",", stringsAsFactors = TRUE)
```

3. Summarize Data

a) Descriptive statistics

```
summary(haiti_ds)
```

```
##                Class            Red            Green             Blue
##  Blue Tarp       : 2022   Min.   : 48   Min.   : 48.0   Min.   : 44.0
##  Rooftop         : 9903   1st Qu.: 80   1st Qu.: 78.0   1st Qu.: 63.0
##  Soil            :20566   Median :163   Median :148.0   Median :123.0
##  Various Non-Tarp: 4744   Mean   :163   Mean   :153.7   Mean   :125.1
##  Vegetation      :26006   3rd Qu.:255   3rd Qu.:226.0   3rd Qu.:181.0
##                           Max.   :255   Max.   :255.0   Max.   :255.0
```

Comment: The haiti_ds has 3 predictors, *red*, *blue*, *green* as colors, with possible values ranging from value 0-255 and discrete variable *class* as the dependent variable.

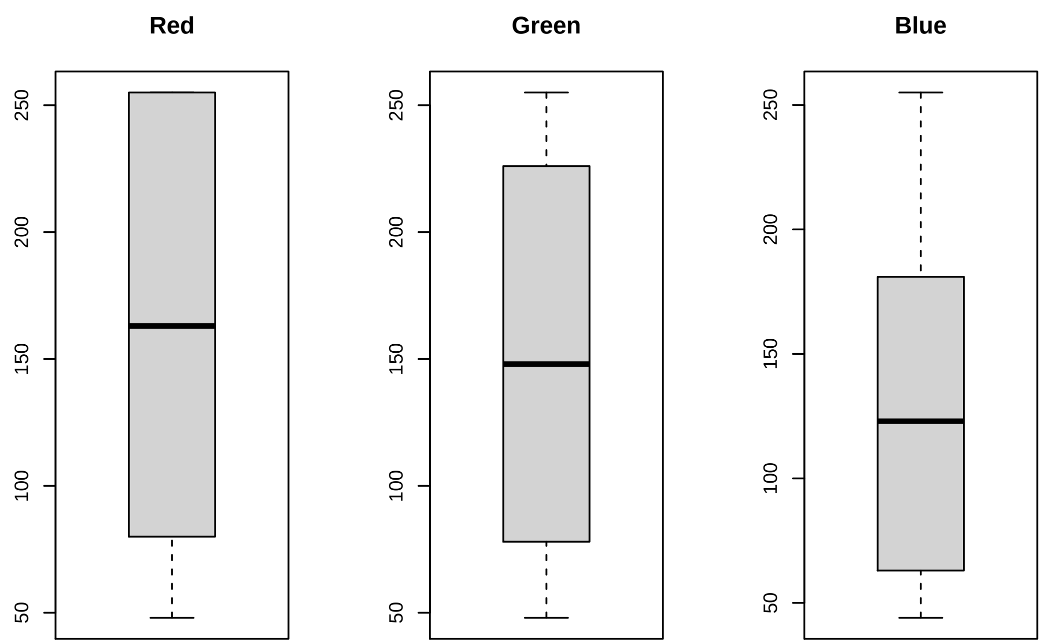
```
percentage <- prop.table(table(haiti_ds$Class)) * 100
cbind(frequency = table(haiti_ds$Class), percentage)
```

```
##                  frequency percentage
## Blue Tarp             2022   3.197293
## Rooftop               9903  15.659145
## Soil                 20566  32.520042
## Various Non-Tarp      4744   7.501463
## Vegetation           26006  41.122057
```

Comment: We can see that the haiti_ds does not have any missing values

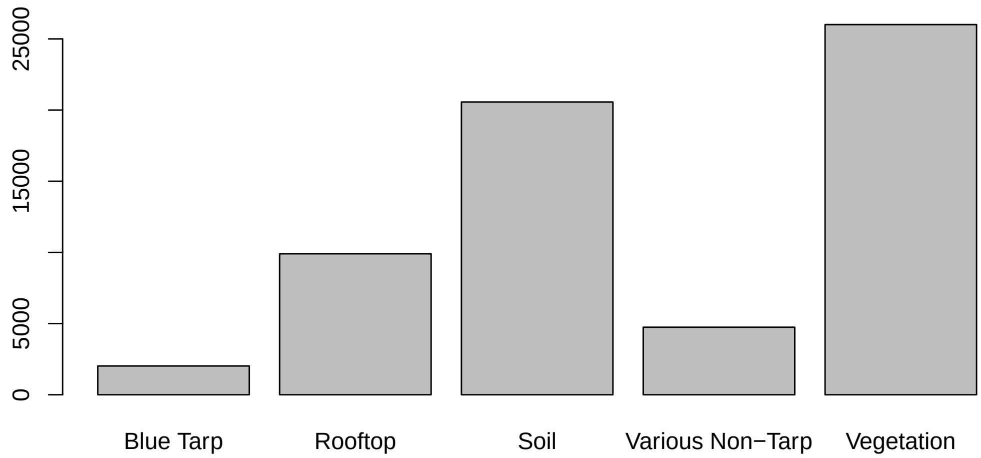
b) Data visualizations

```
# box plot for each of the predictors
par(mfrow = c(1,3))
  for (i in 1:3) {
    boxplot(haiti_ds[,i+1], main= names(haiti_ds)[i+1])
  }
```



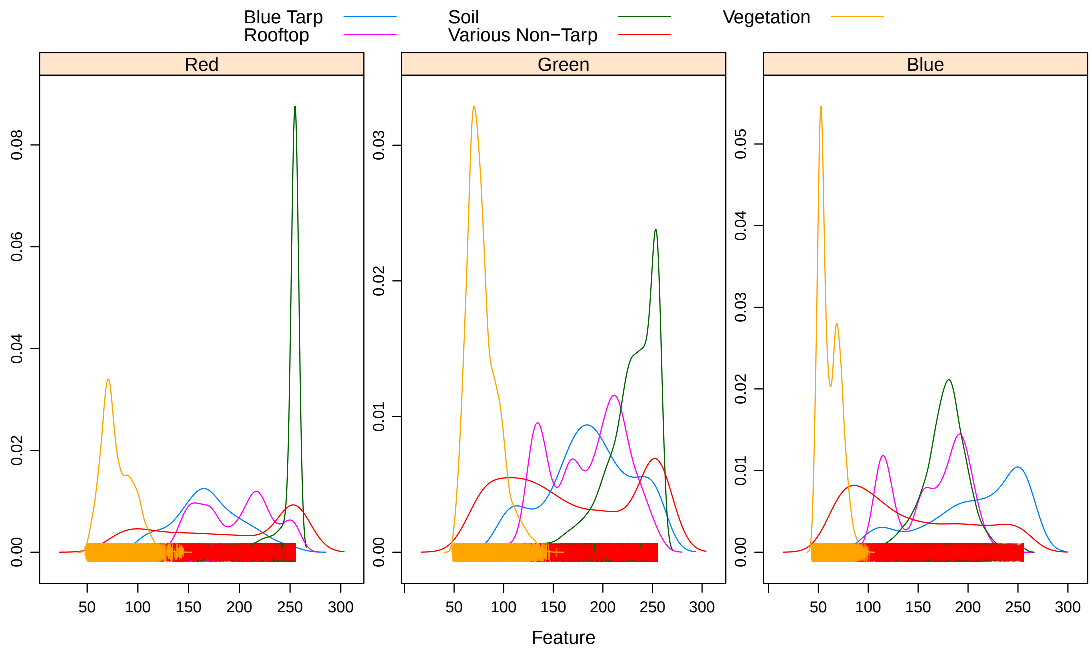
Comment: The above boxplot confirms the data we saw in the above step - haiti_ds summary. The data set is imbalanced.

```r
plot(haiti_ds[,1])
```



Comment: The bar chart confirms the distribution of Class discrete values in the haiti_ds..

```r
# https://www.machinelearningplus.com/machine-learning/caret-package/#4howtovisualizetheimportanceofvar
# density plots for each variable by class value
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=haiti_ds[,2:4], y=haiti_ds$Class, plot="density", scales=scales,adjust = 1.5,
            pch = "|",
            layout = c(3, 1),
            auto.key = list(columns = 3))
```



Comment: We can see from the density plot that Values for BlueTarp class are very mostly normally distributed, but the values for Blue could for this class are right-skewed, which is expected in this case to indicate the blue color tarp.

---

4. Prepare Data

a) check for missing values in the haiti_ds

```
if (sum(is.na(haiti_ds)) > 0) {
  haiti_ds <- na.omit(haiti_ds)
} else {
  print("no missing values in the haiti_ds")
}
```

```
## [1] "no missing values in the haiti_ds"
```

In this study, we are really interested in predicting BlueTarp or Not, and we are not interested in predicting other classes. So we will be creating a new dependent variable called *Class1*. And we will fit different models with Class1 as the response variable.

b) New two-class response variable

```r
#https://r4ds.had.co.nz/transform.html
haiti_ds <-
  mutate(haiti_ds, Class1 = ifelse(haiti_ds$Class == "Blue Tarp",
                                   "BlueTarp",
                                   "NotBlueTarp"))

haiti_ds <-
  mutate(haiti_ds, Class1 = factor(haiti_ds$Class1,
                                   levels = c("NotBlueTarp", "BlueTarp")))

haiti_ds <-
  dplyr::select(haiti_ds, c(Red,Blue,Green,Class1))

# contrasts of Class1 variable
contrasts(haiti_ds$Class1)
```

```
##             BlueTarp
## NotBlueTarp        0
## BlueTarp           1
```

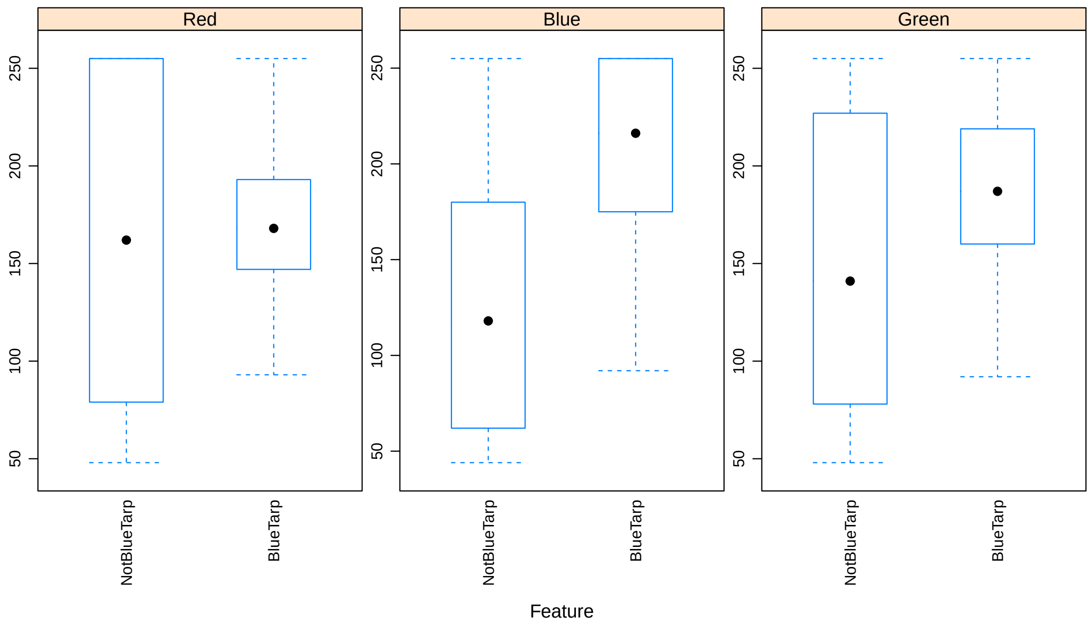
Comment: Add the Class1 dependent categorical variable and drop the *Class* response variable from the original haiti_ds, use the new variable Class1 variable as the dependent variable.

c) Intrunal structure of the dataset

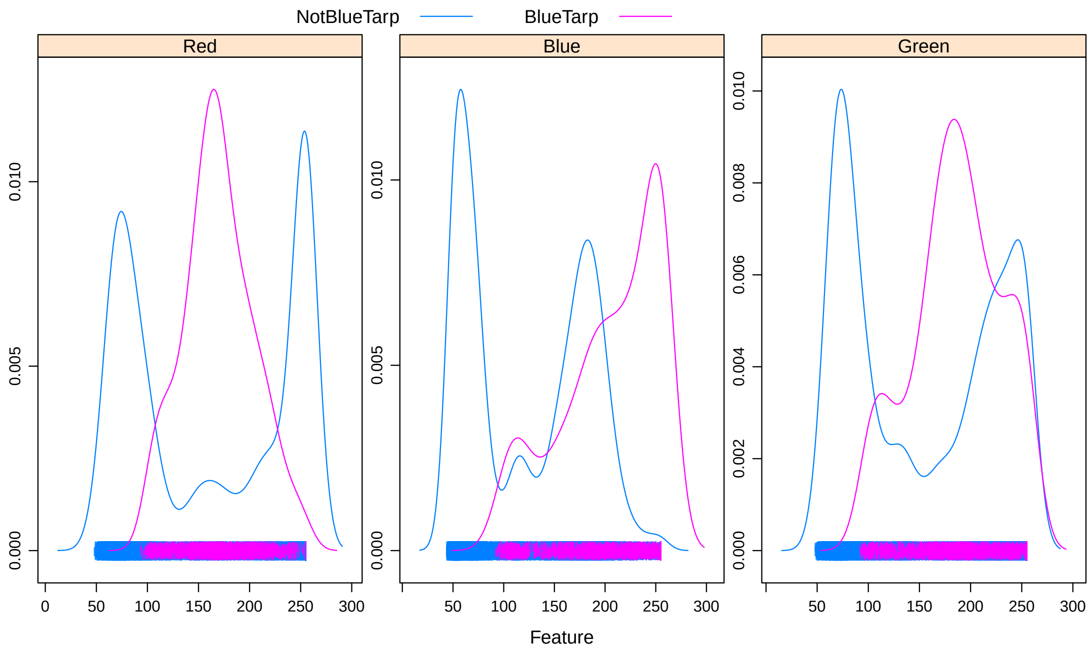
```r
str(haiti_ds)
```

```
## 'data.frame':    63241 obs. of  4 variables:
##  $ Red   : int  64 64 64 75 74 72 71 69 68 67 ...
##  $ Blue  : int  50 50 49 53 54 52 51 49 49 50 ...
##  $ Green : int  67 67 66 82 82 76 72 70 70 70 ...
##  $ Class1: Factor w/ 2 levels "NotBlueTarp",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# https://www.machinelearningplus.com/machine-learning/caret-package/#4howtovisualizetheimportanceofvar
# box and whisker plots for each variable
featurePlot (x = haiti_ds[,1:3],
             y = haiti_ds$Class1,
             plot = "box",
             layout = c(3,1),
             scales = list(y = list(relation ="free"),
                           x = list(rot = 90)),
             auto.key = list(columns = 2))
```




```
# density plots for each variable by class1 value
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=haiti_ds[,1:3], y=haiti_ds$Class1, plot="density", scales=scales,adjust = 1.5,
            pch = "|",
            layout = c(3, 1),
            auto.key = list(columns = 3))
```



Comment: The black dot in the box plots shown is the mean value. For both classes, the red color predictor variable is almost similar, whereas the Green and Blue color predictors have significantly different mean values. Visually at least, this seems to indicate that Blue and Green colors are clearly significant predictors. Although for this study, we will consider all three predictors.

d) Split-out haiti_ds to train and test (validation set)

```
# https://topepo.github.io/caret/data-splitting.html#simple-splitting-with-important-groups
set.seed(0424)
validationIndex <- createDataPartition(haiti_ds$Class1, p = .80, list = FALSE)

# train (and test) haiti_ds - used in CS
train_ds <- haiti_ds[ validationIndex,]

#holdout data set
ho_ds  <- haiti_ds[-validationIndex,]

train_ds <- dplyr::sample_n(train_ds,  nrow(train_ds))
```

Comment: We can see that number of observations with BlueTarp in the Class1 variable is equal to 2022, equal to the number of BlueTarp Class variable observations in supplied haiti_ds, and the number of NotBlueTarp classes is equal to the sum of all other classes found in the haiti_ds.

e) spot-check Class1 distribution for imbalance

Class Frequency distribution in Full haiti_ds dataset:

```
#http://www.u.arizona.edu/~crhummel/FrequencyTable.R
percentage <- prop.table(table(haiti_ds$Class1)) * 100
cbind(frequency = table(haiti_ds$Class1), percentage)
```

```
##             frequency percentage
## NotBlueTarp     61219  96.802707
## BlueTarp         2022   3.197293
```

Class Frequency distribution in train haiti_ds dataset:

```
percentage <- prop.table(table(train_ds$Class1)) * 100
cbind(frequency = table(train_ds$Class1), percentage)
```

```
##             frequency percentage
## NotBlueTarp     48976  96.801992
## BlueTarp         1618   3.198008
```

Class Frequency distribution in holdout haiti_ds dataset:

```
percentage <- prop.table(table(ho_ds$Class1)) * 100
cbind(frequency = table(ho_ds$Class1), percentage)
```

```
##             frequency percentage
## NotBlueTarp     12243  96.805567
## BlueTarp          404   3.194433
```

Comment: We can see that createDataPartition() has crated train and test splits, such that both splits have a similar distribution of the supplied haiti_ds. It confirms that we do not have *imbalance* in the test and train haiti_ds; both *BlueTarp and NotBlueTarp* classes are proportionately represented.

---

5. Evaluate Algorithms

a) setup reusable functions

```
#' Calcuate FDR
#'
#' @param cfmtable - confusion matrix
#'
#' @return FDR value
#'
#' @examples fdr(caret::confusionmatrix$table)
fdr <- function(cfmtable) {
  TN <- cfmtable[1,1]
  TP <- cfmtable[2,2]
  FP <- cfmtable[1,2]
  FN <- cfmtable[2,1]
  return ( FP / (FP+TP))
}
```

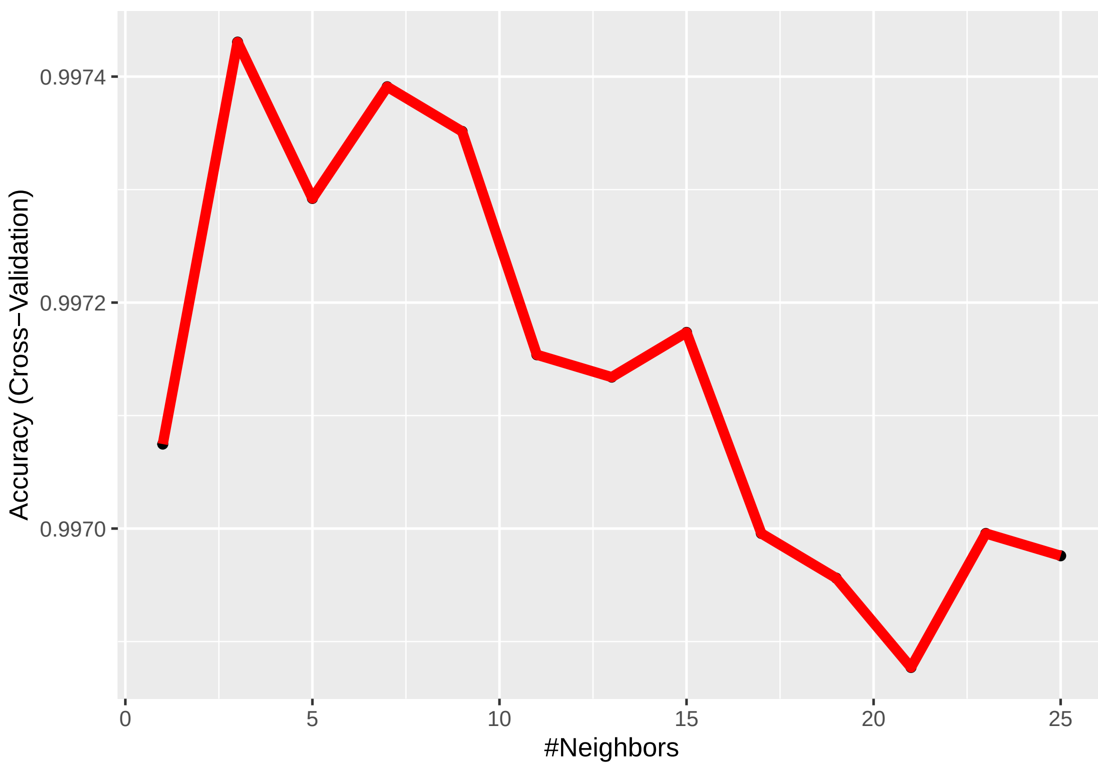
b) Test options and evaluation metric

```
#https://topepo.github.io/caret/model-training-and-tuning.html#control
# test-harness
fitControl <- trainControl(
  method = 'cv',                     # k-fold cross validation
  number = 10,                       # number of folds
  savePredictions = 'final',         # saves predictions for optimal tuning parameter
  classProbs = TRUE,                 # should class probabilities be computed and returned
  #summaryFunction=twoClassSummary, # results summary function
  returnResamp='all'                 # indicator amount resampled summary metrics -
                                     # - saved ("final"/"all"/"none")

  )
#metric
metric <- "Accuracy"
```

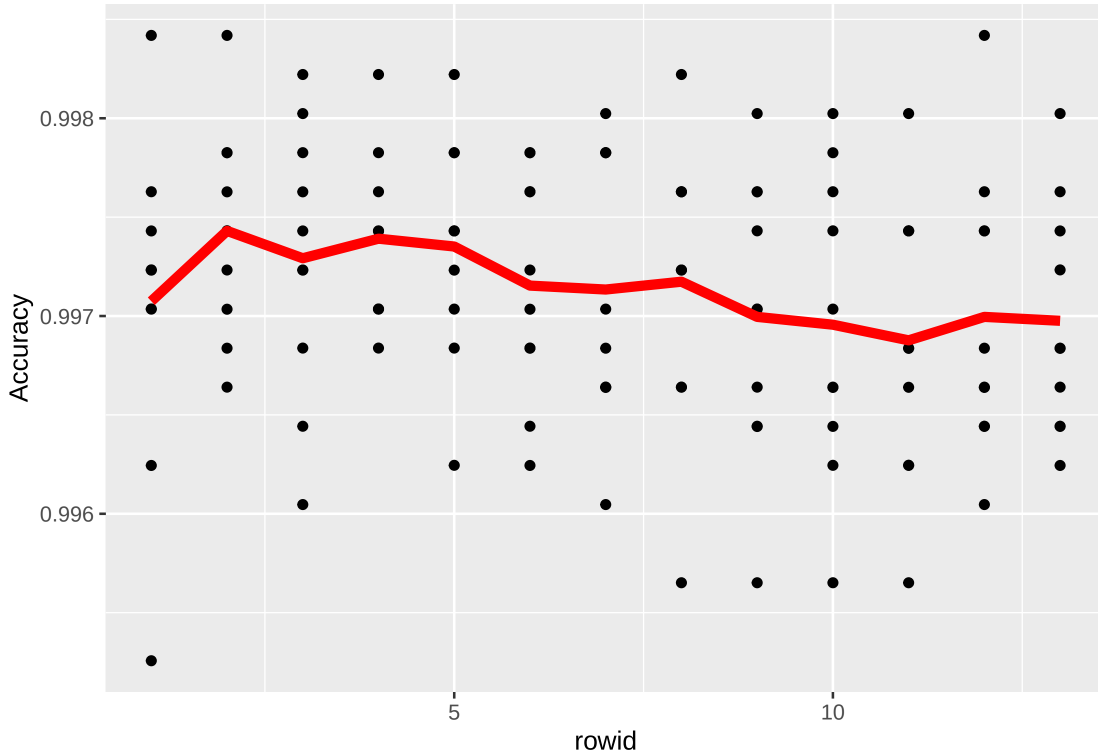
---

b) KNN model

```r
# https://www.machinelearningplus.com/machine-learning/caret-package/#6trainingandtuningthemodel
#KNN
set.seed(seed)
knn_fit <- train (Class1 ~ Blue + Green + Red,
                  data=train_ds,
                  method="knn",
                  preProcess=c("center","scale"),
                  metric = metric,
                  tuneGrid=data.frame(k=seq(1,25,2)),
                  trControl=fitControl)

# live session code example from prof. Scott's lecture
# plot accuracy
knn_fit %>% ggplot(aes(x=seq_along(Accuracy), y=Accuracy)) +
  geom_line(size=2, color='red')
```




```r
# live session code example from prof. Scott's lecture
knn_fit$resample %>%
  dplyr::group_split(Resample) %>%
  purrr::map(rowid_to_column) %>%
  dplyr::bind_rows() %>%
  ggplot(aes(rowid, Accuracy)) + geom_point() +
  geom_smooth(formula='y~x', method='loess', span=.03) +
  geom_line(knn_fit$results, mapping=aes(seq_along(Accuracy), Accuracy),
            size=2, color='red')
```




```r
# predict KNN probabilities
knn_default_raw <- predict(knn_fit, train_ds, type="raw")
knn_default_prob <- predict(knn_fit, train_ds, type="prob")

# create confusion matrix for default threshold (0.5)
knn_default_cfm <- confusionMatrix(reference = train_ds$Class1,
                                   data = knn_default_raw ,
                                   mode='everything',
                                   positive = 'BlueTarp')
knn_default_cfm
```

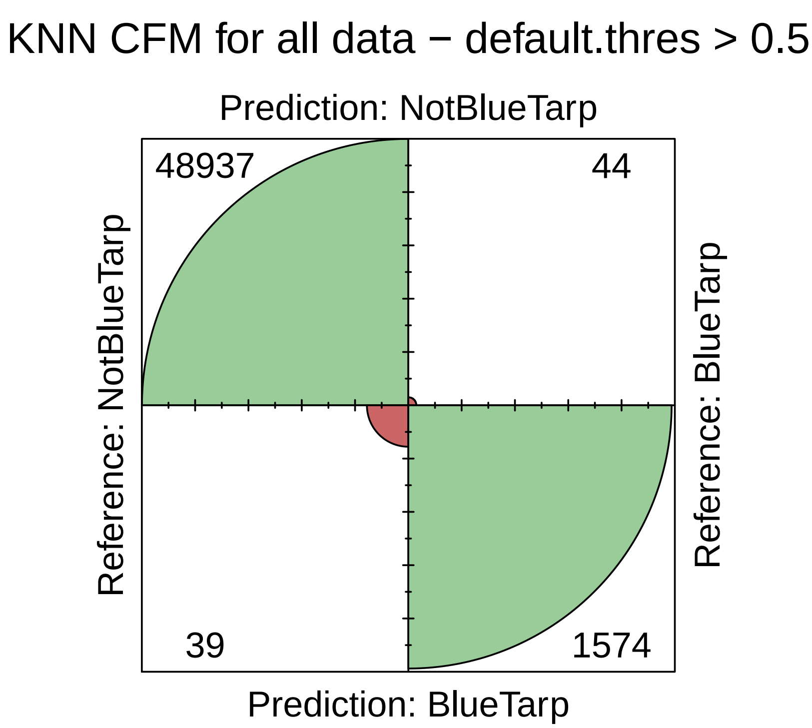
```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction     NotBlueTarp BlueTarp
##   NotBlueTarp       48937       44
##   BlueTarp             39     1574
##
##                   Accuracy : 0.9984
##                     95% CI : (0.998, 0.9987)
##      No Information Rate : 0.968
##      P-Value [Acc > NIR] : <2e-16
##
##                      Kappa : 0.9735
##
##   Mcnemar's Test P-Value : 0.6606
##
##                Sensitivity : 0.97281
```

```
##               Specificity : 0.99920
##            Pos Pred Value : 0.97582
##            Neg Pred Value : 0.99910
##                 Precision : 0.97582
##                    Recall : 0.97281
##                        F1 : 0.97431
##                Prevalence : 0.03198
##            Detection Rate : 0.03111
##      Detection Prevalence : 0.03188
##         Balanced Accuracy : 0.98600
##
##          'Positive' Class : BlueTarp
##
```

```r
predict(knn_fit, type='prob') %>%
  yardstick::roc_auc(truth=train_ds$Class1, "BlueTarp")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary      0.000101
```

```r
# plot confusion matrix for default threshold
fourfoldplot(knn_default_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "KNN CFM for all data - default.thres > 0.5")
```



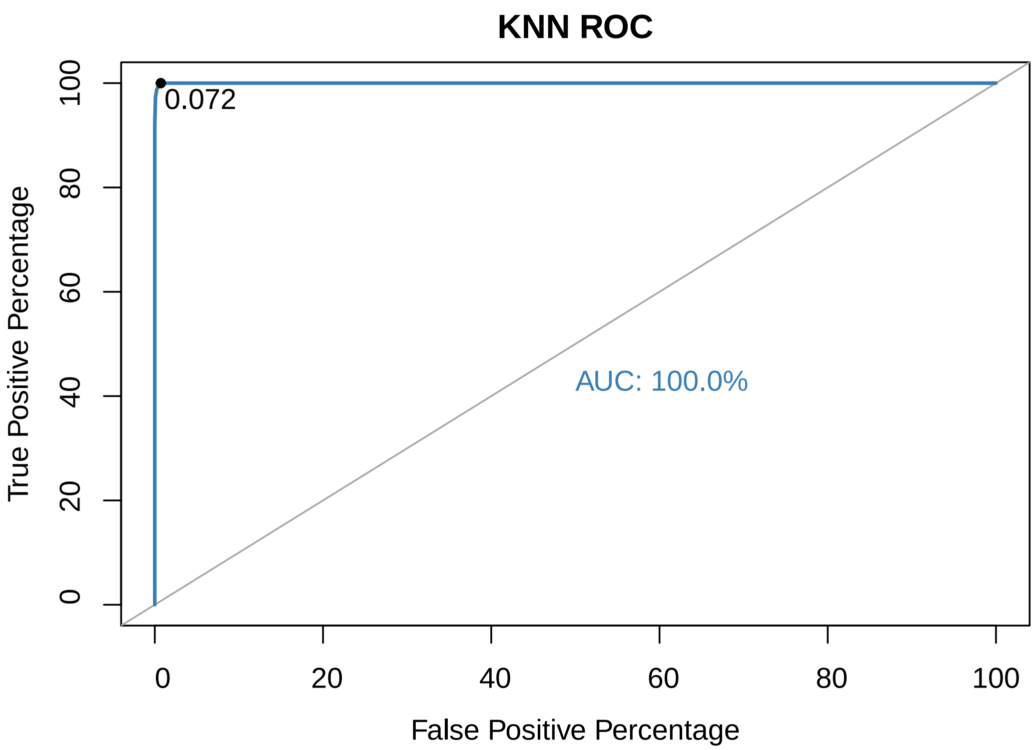
CFM - Confusion Matrix

```r
knn_fit$bestTune
```

```
##   k
## 2 3
```

Comment: #### The best k value for knn is 3

```r
#https://www.youtube.com/watch?v=4jRBRDbJemM&t=615s
knn_roc.info <- roc(train_ds$Class1,knn_default_prob$BlueTarp, main="KNN ROC", col="#377eb8",
                plot=TRUE, legacy.axes =TRUE, asp=NA, percent= TRUE,
                ylab="True Positive Percentage" ,xlab="False Positive Percentage",
                lwd = 2, print.auc=TRUE, print.auc.y=45, print.thres = "best",
                print.thres.pattern="%.3f")
```



Comment: As stated earlier, our goal is to maximize the True positives, that we want to have less false negatives so that more blue tarps, which are blue tarps in the source data, are predicted correctly. We are willing to accept a higher false-positive rate.

The KNN model for the default threshold of 0.5 has 97.22% sensitive, which is already really good. And specificity is also very high at 99.99% We want to consider lowering the threshold so that we can increase the sensitivity of the model.

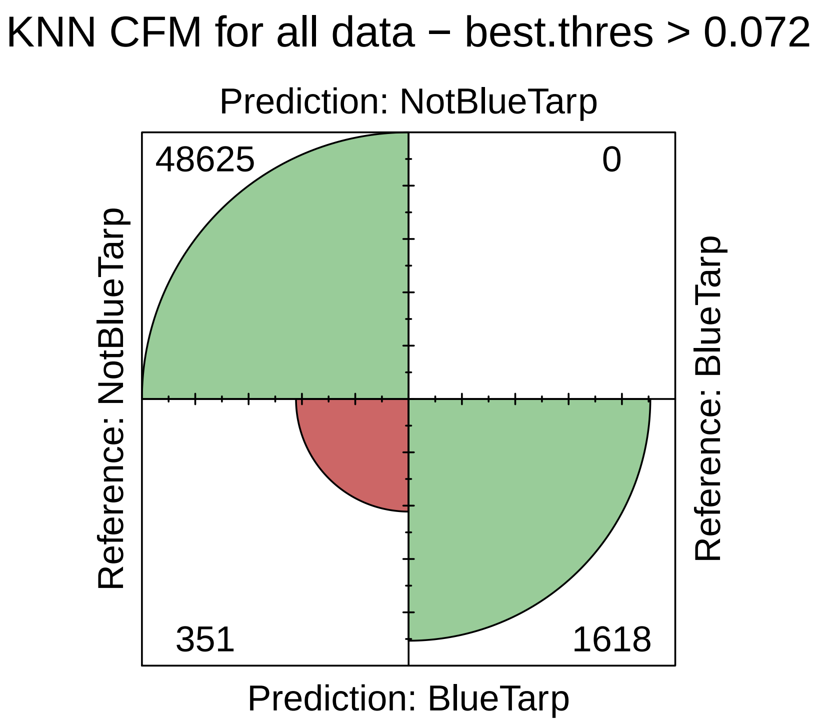
Per ROC curve, we will choose a best threshold value for KNN to be 0.072

```r
# https://www.machinelearningplus.com/machine-learning/caret-package/#65confusionmatrix
# based on roc - select best possble threshold to min
predcted_pred_knn <- as.factor(ifelse(knn_default_prob$BlueTarp > 0.072,
```

```r
                                      'BlueTarp','NotBlueTarp'))

# create confusion matrix for best threshold
knn_thres_cfm <- confusionMatrix(reference = train_ds$Class1,
                                 data = predcted_pred_knn,
                                 mode='everything',
                                 positive = 'BlueTarp')
knn_thres_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       48625        0
##   BlueTarp            351     1618
##
##                Accuracy : 0.9931
##                  95% CI : (0.9923, 0.9938)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8986
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.00000
##             Specificity : 0.99283
##          Pos Pred Value : 0.82174
##          Neg Pred Value : 1.00000
##               Precision : 0.82174
##                  Recall : 1.00000
##                      F1 : 0.90215
##              Prevalence : 0.03198
##          Detection Rate : 0.03198
##    Detection Prevalence : 0.03892
##       Balanced Accuracy : 0.99642
##
##        'Positive' Class : BlueTarp
##
```

```r
# plot confusion matrix for best threshold
fourfoldplot(knn_thres_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "KNN CFM for all data - best.thres > 0.072")
```



CFM - Confusion Matrix

Commnet: We can see that the model sensitivity is increased to 98.89% from 97.22, with a negligible increase in the specificity value.

```r
# predict using the best threshold value for hold out dataset
knn_ho_prob <- predict(knn_fit, ho_ds, type="prob")
knn_ho_pred <- as.factor(ifelse(knn_ho_prob$BlueTarp > 0.072,'BlueTarp','NotBlueTarp'))

# plot confusion matrix for best threshold for hold out dataset
knn_ho_pred_cfm <- confusionMatrix(reference = ho_ds$Class1,
                                   data = knn_ho_pred,
                                   mode='everything',
                                   positive = 'BlueTarp')

knn_ho_pred_cfm
```

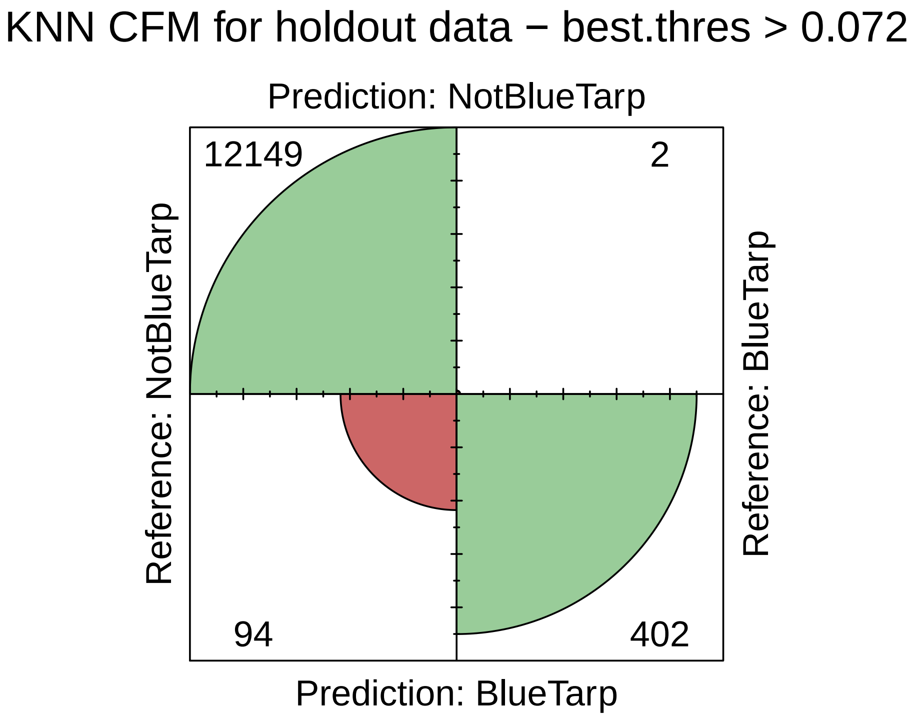
```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       12149        2
##   BlueTarp             94      402
##
##                Accuracy : 0.9924
##                  95% CI : (0.9907, 0.9938)
##     No Information Rate : 0.9681
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.8894
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.99505
##             Specificity : 0.99232
##          Pos Pred Value : 0.81048
##          Neg Pred Value : 0.99984
##               Precision : 0.81048
##                  Recall : 0.99505
##                      F1 : 0.89333
##              Prevalence : 0.03194
##          Detection Rate : 0.03179
##    Detection Prevalence : 0.03922
##       Balanced Accuracy : 0.99369
##
##        'Positive' Class : BlueTarp
##
```

```r
knn.fdr <- fdr(knn_ho_pred_cfm$table)
knn.fdr
```

```
## [1] 0.004950495
```

```r
# https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/fourfoldplot
# plot confusion matrix for best threshold for hold out dataset
fourfoldplot(knn_ho_pred_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "KNN CFM for holdout data - best.thres > 0.072")
```



CFM - Confusion Matrix

---

c) LDA model accuracy estimate

```
#LDA
set.seed(seed)
lda_fit <- train(Class1 ~  Blue + Green + Red,
                 data=train_ds,
                 method="lda",
                 metric=metric,
                 preProcess=c("center","scale"),
                 trControl=fitControl)

lda_fit
```

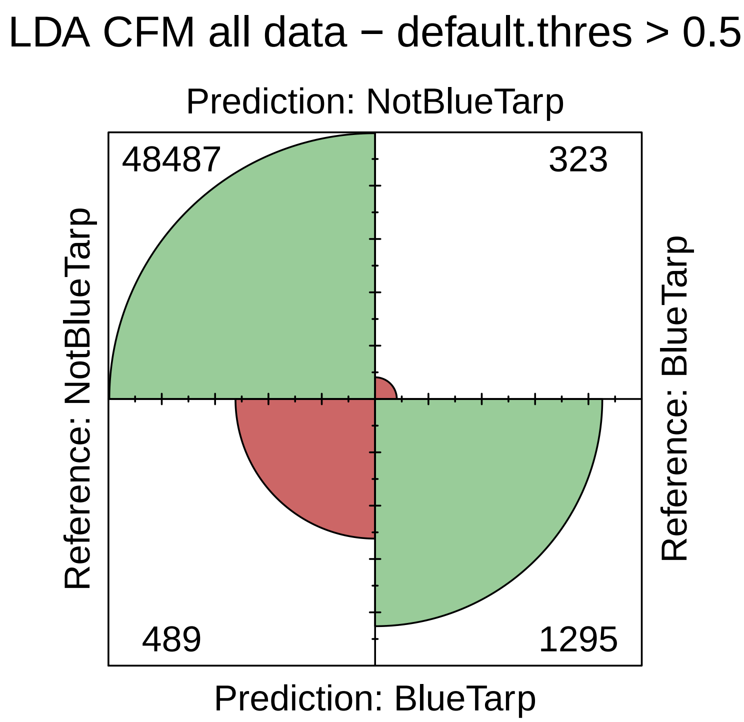
```
## Linear Discriminant Analysis
##
## 50594 samples
##     3 predictor
##     2 classes: 'NotBlueTarp', 'BlueTarp'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 45535, 45535, 45535, 45534, 45536, 45534, ...
```

```
## Resampling results:
##
##   Accuracy   Kappa
##   0.9839506  0.7530352
```

```
# predict LDA probabilities
lda_default_raw <- predict(lda_fit, train_ds, type="raw")
lda_default_prob <- predict(lda_fit, train_ds, type="prob")

# create confusion matrix for default threshold (0.5)
lda_default_cfm <- confusionMatrix(reference = train_ds$Class1,
                                   data = lda_default_raw,
                                   mode='everything',
                                   positive = 'BlueTarp')
lda_default_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       48487      323
##   BlueTarp            489     1295
##
##                Accuracy : 0.984
##                  95% CI : (0.9828, 0.985)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.753
##
##  Mcnemar's Test P-Value : 7.023e-09
##
##             Sensitivity : 0.80037
##             Specificity : 0.99002
##          Pos Pred Value : 0.72590
##          Neg Pred Value : 0.99338
##               Precision : 0.72590
##                  Recall : 0.80037
##                      F1 : 0.76132
##              Prevalence : 0.03198
##          Detection Rate : 0.02560
##    Detection Prevalence : 0.03526
##       Balanced Accuracy : 0.89519
##
##        'Positive' Class : BlueTarp
##
```

```
# plot confusion matrix for default threshold
fourfoldplot(lda_default_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "LDA CFM all data - default.thres > 0.5")
```

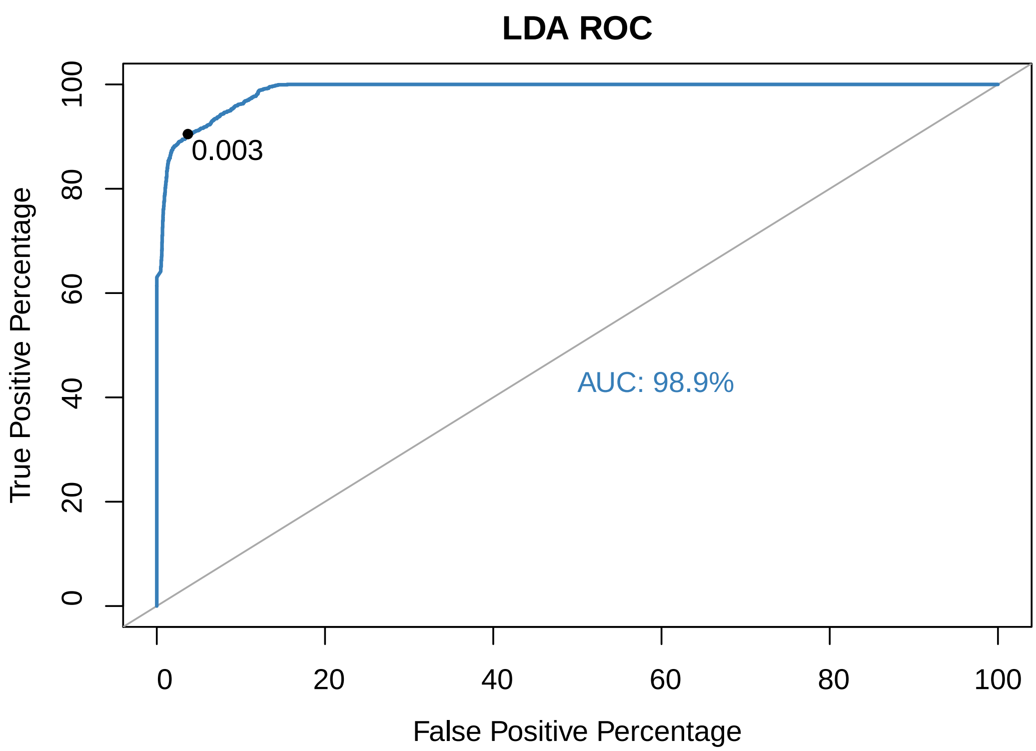


CFM - Confusion Matrix

```
# roc and auc
# https://en.wikipedia.org/wiki/F1_score
# https://stackoverflow.com/questions/57183675/proc-package-with-pre-specified-cutoff-values-with-two-d
auc(train_ds$Class1,lda_default_prob$BlueTarp )
```

```
## Area under the curve: 0.9885
```

```
lda.roc.info <- roc(train_ds$Class1,lda_default_prob$BlueTarp, main="LDA ROC", col="#377eb8",
                plot=TRUE, legacy.axes =TRUE, asp=NA, percent= TRUE,
                ylab="True Positive Percentage" ,xlab="False Positive Percentage",
                lwd = 2, print.auc=TRUE, print.auc.y=45, print.thres = "best",
                print.thres.pattern="%.3f")
```



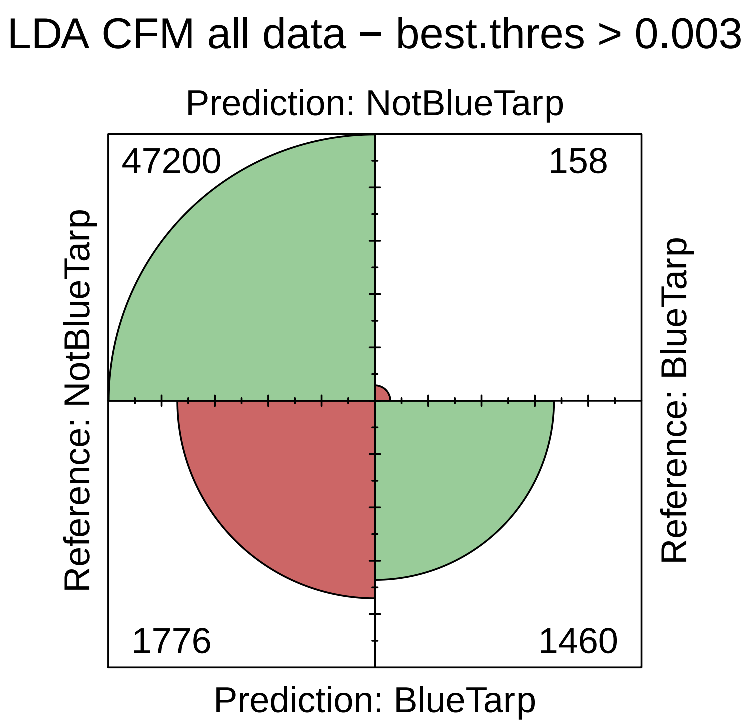
Per ROC curve, we will choose a best threshold value for LDA to be 0.003

```
# based on roc - select best possble threshold to min
predcted_pred_lda <- as.factor(ifelse(lda_default_prob$BlueTarp > 0.003,
                                      'BlueTarp',
                                      'NotBlueTarp'))

# create confusion matrix for best threshold
lda_thres_cfm <- confusionMatrix(reference = train_ds$Class1,
                                 data = predcted_pred_lda,
                                 mode='everything',
                                 positive = 'BlueTarp')
lda_thres_cfm
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction     NotBlueTarp BlueTarp
##   NotBlueTarp       47200      158
##   BlueTarp           1776     1460
##
##                Accuracy : 0.9618
##                  95% CI : (0.9601, 0.9634)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.5838
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.90235
##             Specificity : 0.96374
##          Pos Pred Value : 0.45117
##          Neg Pred Value : 0.99666
##               Precision : 0.45117
##                  Recall : 0.90235
##                      F1 : 0.60157
##              Prevalence : 0.03198
##          Detection Rate : 0.02886
##    Detection Prevalence : 0.06396
##       Balanced Accuracy : 0.93304
##
##        'Positive' Class : BlueTarp
##
```

```
# plot confusion matrix for best threshold
fourfoldplot(lda_thres_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "LDA CFM all data - best.thres > 0.003")
```



CFM - Confusion Matrix

```
# predict using the best threshold value for hold out dataset
lda_ho_prob <- predict(lda_fit, ho_ds, type="prob")
lda_ho_pred <- as.factor(ifelse(lda_ho_prob$BlueTarp > 0.003,'BlueTarp','NotBlueTarp'))
```

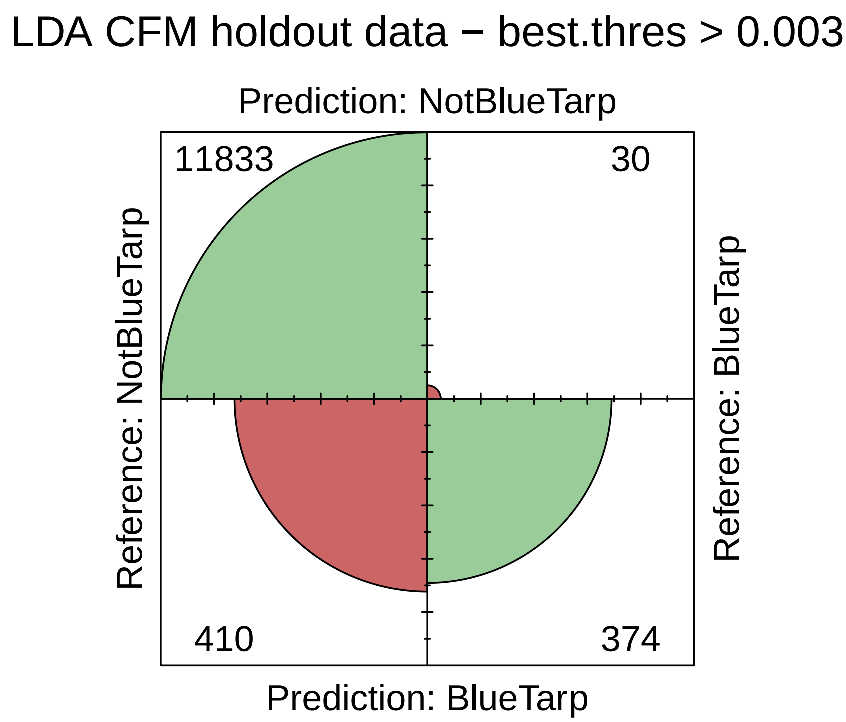
```r
# plot confusion matrix for best threshold for hold out dataset
lda_ho_pred_cfm <- confusionMatrix(reference = ho_ds$Class1,
                                   data = lda_ho_pred,
                                   mode='everything',
                                   positive = 'BlueTarp')
lda_ho_pred_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       11833       30
##   BlueTarp            410      374
##
##                Accuracy : 0.9652
##                  95% CI : (0.9619, 0.9683)
##     No Information Rate : 0.9681
##     P-Value [Acc > NIR] : 0.9662
##
##                   Kappa : 0.6133
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.92574
##             Specificity : 0.96651
##          Pos Pred Value : 0.47704
##          Neg Pred Value : 0.99747
##               Precision : 0.47704
##                  Recall : 0.92574
##                      F1 : 0.62963
##              Prevalence : 0.03194
##          Detection Rate : 0.02957
##    Detection Prevalence : 0.06199
##       Balanced Accuracy : 0.94613
##
##        'Positive' Class : BlueTarp
##
```

```r
lda.fdr <- fdr(lda_ho_pred_cfm$table)
lda.fdr
```

```
## [1] 0.07425743
```

```r
# plot confusion matrix for best threshold for hold out dataset
fourfoldplot(lda_ho_pred_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "LDA CFM holdout data - best.thres > 0.003")
```



CFM - Confusion Matrix

Note : References shown till now are the references are the same references used in the code following these sections.

---

d) QDA model accuracy estimate

```
#QDA
set.seed(seed)
qda_fit <- train(Class1 ~  Blue + Green + Red,
                 data=train_ds,
                 method="qda",
                 metric=metric,
                 preProcess=c("center","scale"),
                 trControl=fitControl)

qda_fit
```

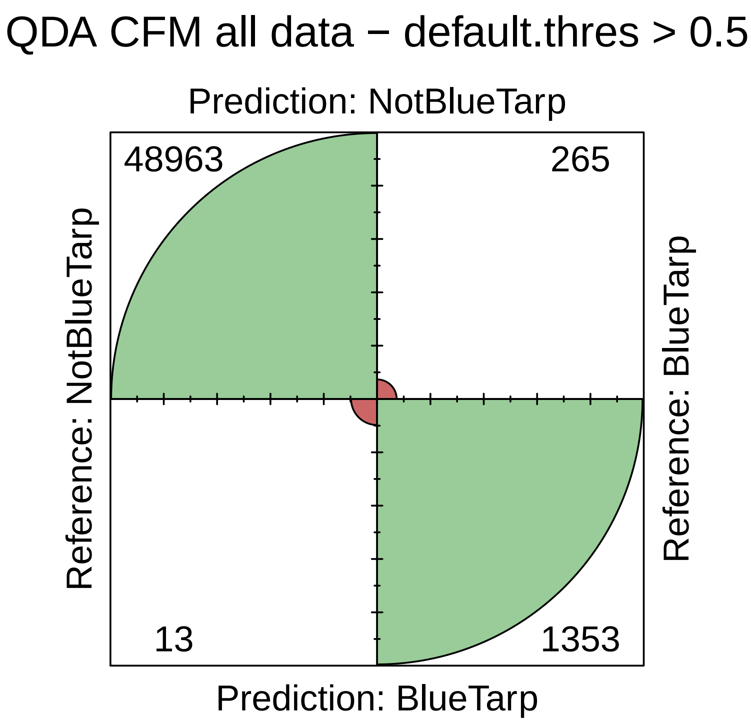
```
## Quadratic Discriminant Analysis
##
## 50594 samples
##     3 predictor
##     2 classes: 'NotBlueTarp', 'BlueTarp'
##
```

```
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 45535, 45535, 45535, 45534, 45536, 45534, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9945052  0.9036955
```

```
# predict qda probabilities
qda_default_raw <- predict(qda_fit, train_ds, type="raw")
qda_default_prob <- predict(qda_fit, train_ds, type="prob")

# create confusion matrix for default threshold (0.5)
qda_default_cfm <- confusionMatrix(reference = train_ds$Class1,
                                   data = qda_default_raw,
                                   mode='everything',
                                   positive = 'BlueTarp')
qda_default_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       48963      265
##   BlueTarp             13     1353
##
##                Accuracy : 0.9945
##                  95% CI : (0.9938, 0.9951)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.904
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.83622
##             Specificity : 0.99973
##          Pos Pred Value : 0.99048
##          Neg Pred Value : 0.99462
##               Precision : 0.99048
##                  Recall : 0.83622
##                      F1 : 0.90684
##              Prevalence : 0.03198
##          Detection Rate : 0.02674
##    Detection Prevalence : 0.02700
##       Balanced Accuracy : 0.91798
##
##        'Positive' Class : BlueTarp
##
```

```
# plot confusion matrix for default threshold
fourfoldplot(qda_default_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "QDA CFM all data - default.thres > 0.5")
```

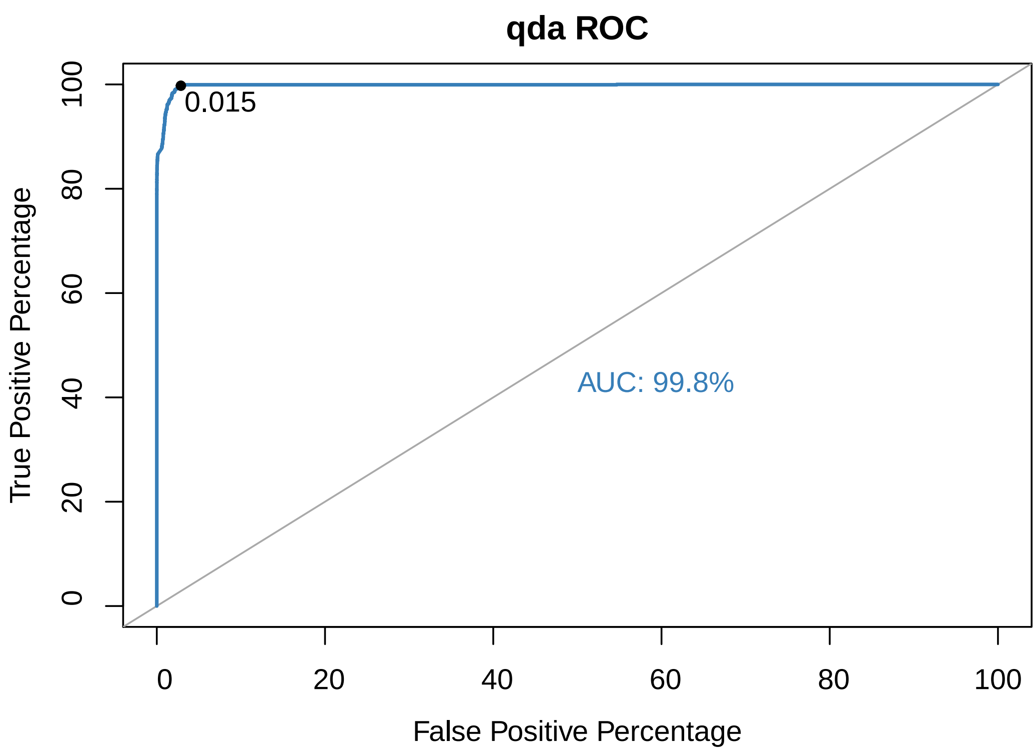


CFM - Confusion Matrix

```
# roc and auc
auc(train_ds$Class1,qda_default_prob$BlueTarp )
```

```
## Area under the curve: 0.9981
```

```
qda.roc.info <- roc(train_ds$Class1,qda_default_prob$BlueTarp, main="qda ROC", col="#377eb8",
                plot=TRUE, legacy.axes =TRUE, asp=NA, percent= TRUE,
                ylab="True Positive Percentage" ,xlab="False Positive Percentage",
                lwd = 2, print.auc=TRUE, print.auc.y=45, print.thres = "best",
                print.thres.pattern="%.3f")
```



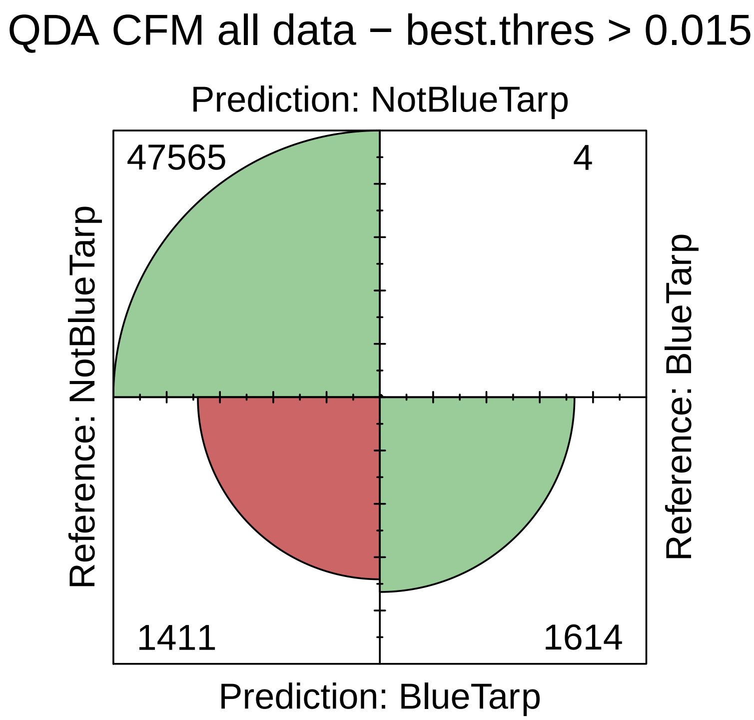
Per ROC curve, we will choose a best threshold value for QDA to be 0.015

```
# based on roc - select best possble threshold to min
predcted_pred_qda <- as.factor(ifelse(qda_default_prob$BlueTarp > 0.015,
                                      'BlueTarp',
                                      'NotBlueTarp'))

# create confusion matrix for best threshold
qda_thres_cfm <- confusionMatrix(reference = train_ds$Class1,
                                 data = predcted_pred_qda,
                                 mode='everything',
                                 positive = 'BlueTarp')
qda_thres_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       47565        4
##   BlueTarp           1411     1614
##
##                Accuracy : 0.972
##                  95% CI : (0.9706, 0.9735)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 8.862e-08
##
##                   Kappa : 0.682
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.99753
##             Specificity : 0.97119
##          Pos Pred Value : 0.53355
##          Neg Pred Value : 0.99992
##               Precision : 0.53355
##                  Recall : 0.99753
##                      F1 : 0.69524
##              Prevalence : 0.03198
##          Detection Rate : 0.03190
##    Detection Prevalence : 0.05979
##       Balanced Accuracy : 0.98436
##
##        'Positive' Class : BlueTarp
##
```

```
# plot confusion matrix for best threshold
fourfoldplot(qda_thres_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "QDA CFM all data - best.thres > 0.015")
```



CFM - Confusion Matrix

```
# predict using the best threshold value for hold out dataset
qda_ho_prob <- predict(qda_fit, ho_ds, type="prob")
qda_ho_pred <- as.factor(ifelse(qda_ho_prob$BlueTarp > 0.015,'BlueTarp','NotBlueTarp'))
```

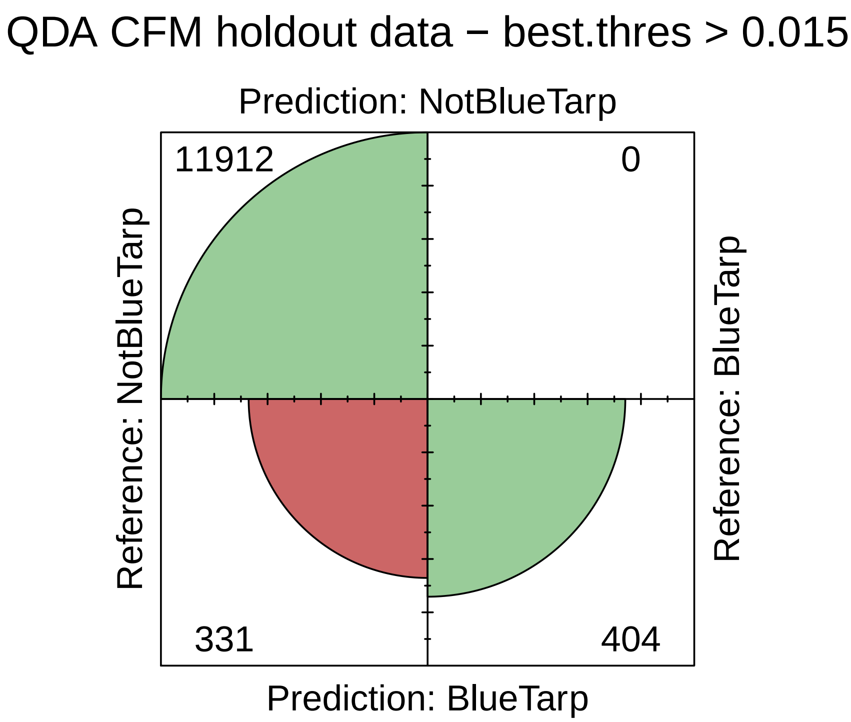
```r
# plot confusion matrix for best threshold for hold out dataset
qda_ho_pred_cfm <- confusionMatrix(reference = ho_ds$Class1,
                                   data = qda_ho_pred,
                                   mode='everything',
                                   positive = 'BlueTarp')
qda_ho_pred_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       11912        0
##   BlueTarp            331      404
##
##                Accuracy : 0.9738
##                  95% CI : (0.9709, 0.9765)
##     No Information Rate : 0.9681
##     P-Value [Acc > NIR] : 8.072e-05
##
##                   Kappa : 0.6969
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 1.00000
##             Specificity : 0.97296
##          Pos Pred Value : 0.54966
##          Neg Pred Value : 1.00000
##               Precision : 0.54966
##                  Recall : 1.00000
##                      F1 : 0.70939
##              Prevalence : 0.03194
##          Detection Rate : 0.03194
##    Detection Prevalence : 0.05812
##       Balanced Accuracy : 0.98648
##
##        'Positive' Class : BlueTarp
##
```

```r
qda.fdr <- fdr(qda_ho_pred_cfm$table)
qda.fdr
```

```
## [1] 0
```

```r
# plot confusion matrix for best threshold for hold out dataset
fourfoldplot(qda_ho_pred_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "QDA CFM holdout data - best.thres > 0.015")
```



CFM - Confusion Matrix

---

e) LR model accuracy estimate

```
#LR
set.seed(seed)
lr_fit <- train(Class1 ~  Blue + Green + Red,
                data=train_ds,
                method="glm",
                metric=metric,
                    family ="binomial",
                preProcess=c("center","scale"),
                trControl=fitControl)

lr_fit
```

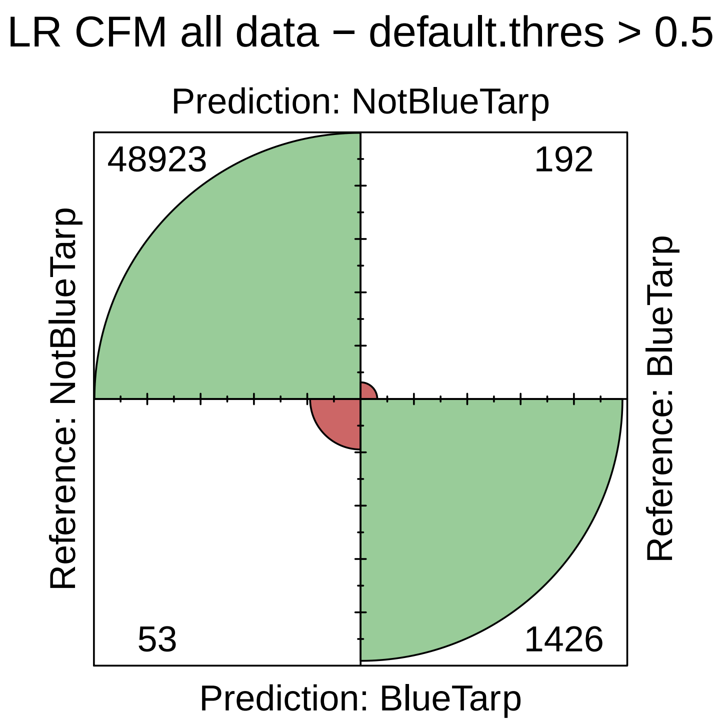
```
## Generalized Linear Model
##
## 50594 samples
##     3 predictor
##     2 classes: 'NotBlueTarp', 'BlueTarp'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 45535, 45535, 45535, 45534, 45536, 45534, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9951377  0.9178006
```

```
# predict lr probabilities
lr_default_raw <- predict(lr_fit, train_ds, type="raw")
lr_default_prob <- predict(lr_fit, train_ds, type="prob")

# create confusion matrix for default threshold (0.5)
lr_default_cfm <- confusionMatrix(reference = train_ds$Class1,
                                   data = lr_default_raw,
                                   mode='everything',
                                   positive = 'BlueTarp')
lr_default_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       48923      192
##   BlueTarp             53     1426
##
##                Accuracy : 0.9952
##                  95% CI : (0.9945, 0.9957)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9184
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.88133
##             Specificity : 0.99892
##          Pos Pred Value : 0.96416
##          Neg Pred Value : 0.99609
##               Precision : 0.96416
##                  Recall : 0.88133
##                      F1 : 0.92089
##              Prevalence : 0.03198
##          Detection Rate : 0.02819
##    Detection Prevalence : 0.02923
##       Balanced Accuracy : 0.94013
##
##        'Positive' Class : BlueTarp
##
```

```
# plot confusion matrix for default threshold
fourfoldplot(lr_default_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "LR CFM all data - default.thres > 0.5")
```

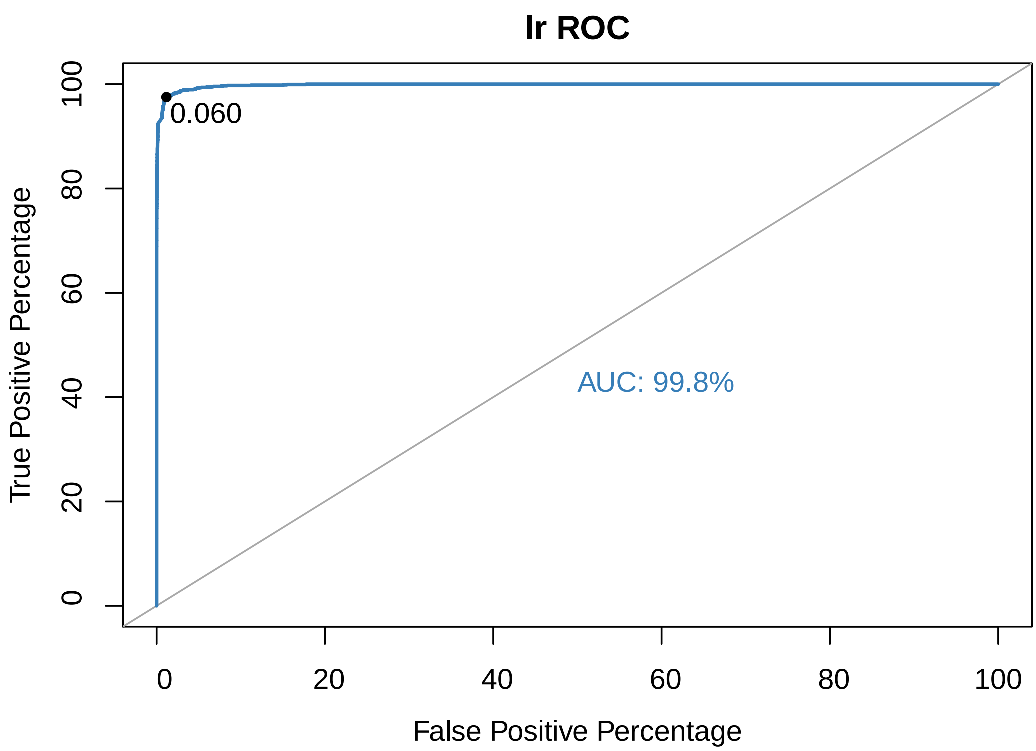


CFM - Confusion Matrix

```
# roc and auc
auc(train_ds$Class1,lr_default_prob$BlueTarp )
```

```
## Area under the curve: 0.9983
```

```
lr.roc.info <- roc(train_ds$Class1,lr_default_prob$BlueTarp, main="lr ROC", col="#377eb8",
                plot=TRUE, legacy.axes =TRUE, asp=NA, percent= TRUE,
                ylab="True Positive Percentage" ,xlab="False Positive Percentage",
                lwd = 2, print.auc=TRUE, print.auc.y=45, print.thres = "best",
                print.thres.pattern="%.3f")
```



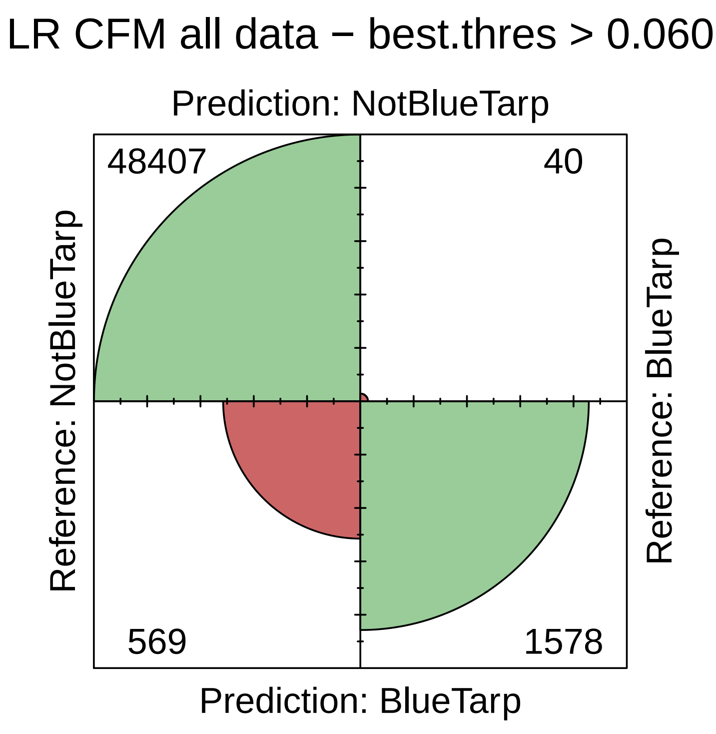
Per ROC curve, we will choose a best threshold value for LR to be 0.060

```
# based on roc - select best possble threshold to min
predcted_pred_lr <- as.factor(ifelse(lr_default_prob$BlueTarp > 0.060,
                                      'BlueTarp',
                                      'NotBlueTarp'))

# create confusion matrix for best threshold
lr_thres_cfm <- confusionMatrix(reference = train_ds$Class1,
                                 data = predcted_pred_lr,
                                 mode='everything',
                                 positive = 'BlueTarp')
lr_thres_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       48407       40
##   BlueTarp            569     1578
##
##                Accuracy : 0.988
##                  95% CI : (0.987, 0.9889)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8321
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.97528
##             Specificity : 0.98838
##          Pos Pred Value : 0.73498
##          Neg Pred Value : 0.99917
##               Precision : 0.73498
##                  Recall : 0.97528
##                      F1 : 0.83825
##              Prevalence : 0.03198
##          Detection Rate : 0.03119
##    Detection Prevalence : 0.04244
##       Balanced Accuracy : 0.98183
##
##        'Positive' Class : BlueTarp
##
```

```
# plot confusion matrix for best threshold
fourfoldplot(lr_thres_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "LR CFM all data - best.thres > 0.060")
```



CFM - Confusion Matrix

```
# predict using the best threshold value for hold out dataset
lr_ho_prob <- predict(lr_fit, ho_ds, type="prob")
lr_ho_pred <- as.factor(ifelse(lr_ho_prob$BlueTarp > 0.060,'BlueTarp','NotBlueTarp'))
```

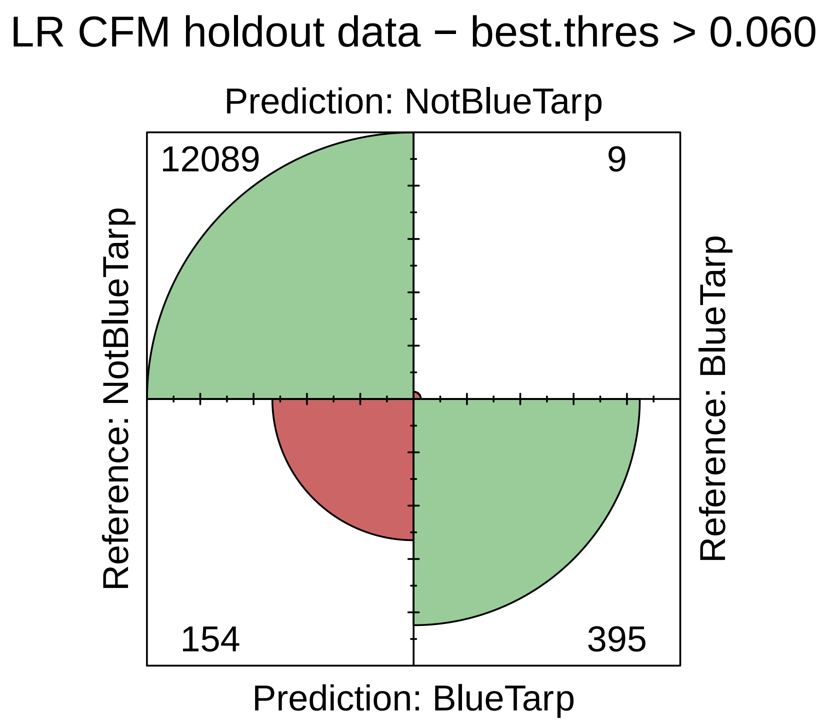
```r
# plot confusion matrix for best threshold for hold out dataset
lr_ho_pred_cfm <- confusionMatrix(reference = ho_ds$Class1,
                                  data = lr_ho_pred,
                                  mode='everything',
                                  positive = 'BlueTarp')
lr_ho_pred_cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    NotBlueTarp BlueTarp
##   NotBlueTarp       12089        9
##   BlueTarp            154      395
##
##                Accuracy : 0.9871
##                  95% CI : (0.985, 0.989)
##     No Information Rate : 0.9681
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8224
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.97772
##             Specificity : 0.98742
##          Pos Pred Value : 0.71949
##          Neg Pred Value : 0.99926
##               Precision : 0.71949
##                  Recall : 0.97772
##                      F1 : 0.82896
##              Prevalence : 0.03194
##          Detection Rate : 0.03123
##    Detection Prevalence : 0.04341
##       Balanced Accuracy : 0.98257
##
##        'Positive' Class : BlueTarp
##
```

```r
lr.fdr <- fdr(lr_ho_pred_cfm$table)
lr.fdr
```

```
## [1] 0.02227723
```

```r
# plot confusion matrix for best threshold for hold out dataset
fourfoldplot(lr_ho_pred_cfm$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1,
             main = "LR CFM holdout data - best.thres > 0.060")
```



CFM - Confusion Matrix

6. Finalize Model

a) K-Folds Out of Sampling Performance

Method	Knn (k=3)	LDA	QDA	Logistic Regression
Accuracy	99.24%	96.52%	97.38%	98.71%
AUC	100.00%	98.90%	99.80%	99.80%
ROC	✓	✓	✓	✓
Threshold	0.072	0.003	0.015	0.060
Sensitivty=Recall=Power	99.51%	92.57%	100.00%	97.77%
Spectitivty=1-FPR	99.23%	96.65%	97.30%	98.74%
FDR	0.50%	7.43%	0.00%	2.23%
Precision=PPV	81.05%	47.70%	54.97%	71.95%

Table 1: Performance Metrics : 10-Fold Cross-Validation Metrics

Figure 1: K-Folds Out of Sampling Performance

Note: Scores are attached from excell spread-sheet

b) Best Algorithm

Q1. KNN

7. Conclusion

a). Which algorithm works best?

In this project, as per the project's goal, we are interested in finding an effective algorithm to predict more blue tarps correctly identified as blue tarps, so that rescue workers can help more people who needed it. i.e., we are interested in how many blue tarps were correctly identified as blue tarps?

Criteria for choosing the best algorithm:

- *Accuracy* tells us that out of all classes, how many were predicted correctly, we want this to be as high as possible so that Blue tarps that were blue tarps are predicted as blue tarps, and no blue tarp image (NoBlueTarp) is predicted as no blue tarp image.
- *Sensitivity* tells us how many items were correctly selected as blue tarps (positive class) that were actually blue tarp images.
- *Precision* tells us out of actual blue tarp images how many were correctly predicted as blue tarps.
- *Specificity* gives us the proportions of images - no blue tarps(NotBlueTarp) were correct classified as not blue tarps.

We also know that resources are limited in a rescue operation and should not go to waste, so we want to find an effective algorithm that maximizes the sensitivity; we want to be biased towards high sensitivity to direct rescue workers correctly to as people as possible while striving maimizing specitivity.

Reviewing the results tabulated above shown k-fold out of sampling performance table, the KNN is the best performing model as it has the highest -

- Accuracy =99.24%,
- Sensitivity =99.51%,
- Specificity = 99.23%, &
- Precision = 81.05%,

As we can see in section 5, we could get the best performance out of KNN after fitting the data again with the best threshold value of 0.072.
LR was the second-best performing model with a precision of 71.95%, followed by QDA with 54.97% precision, and LDA has the lowest precision

b). Justification for choosing threshold value

https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

The given image dataset is highly imbalanced, with only 3.20% blue tarps, and the rest is 96.80%, not blue tarps.

A default threshold of 0.5 may not represent an optimal interpretation of predicted probabilities or scoring into a class. In such situations, changing the threshold value from the default value of 0.5 is one of the proven techniques of effectively handling class imbalance.

As noted above, our goal is to have a model that has high sensitivity and high precision. So we want to reduce the threshold to a value less than the default 0.5. Each model's threshold was selected based on the best threshold value shown on the corresponding ROC curves.

c). Other adaquate performing models

Were there multiple adequately performing methods, or just one clear best method?

In this study, even though the dataset was imbalanced with the best threshold, the KNN model can effectively classify both blue tarp (BlueTarp) and not blue tarp (NotBlueTarp) images.

However, the other models LDA, QDA, and Logistic Regression (LR) model results can be further improved,
- 1. Source more balanced dataset
- 2. Apply other resampling techniques such as bootstrap and leave one out cross-validation (LOOCV) methods - 3. We can try penalized models
- 4. We try differnt threshold values other than suggested by ROC curve

d). Other adequately performing methods

Were there multiple adequately performing methods, or just one clear best method?

The logistic regression model was the second-best performing model with -
- *Accuracy =99.81%,*
- *Sensitivity =97.80%,*
- *Specificity = 98.74%, &*
- *Precision = 71.05%,*

Both LDA and QDA has low precision will not meet the goal of effectively finding blue tarps images. These models have high false positives, which will lead to wasted resources if the rescue workers were to drop much-needed resources to locations that were misclassified as blue tarps when in actuality, they are not blue tarps.

e) Data forumlation

What is it about this data formulation that allows us to address it with predictive modeling tools?

The observations(records) in given the data set represent an image using the RGB color model. The RGB color model's main purpose is for the sensing, representation, and display of images in electronic systems. The source data has 5 different classes represented as the response variable and Red, Green, and Blue as three predictor variables representing the RGB model used to classify each image into 5 different classes. Our study, since we are only interested in predicting blue tarps and not blue tarps, introduced a new response variable class1 with only two classes - BlueTarp and NotBlueTarp. The three predictors are continuous variables with the same range of values (0 to 255). Hence this dataset is well suited for binary classification.

e) Effectiveness this study for saving human life

How effective do you think your work here could actually be in terms of helping to save human life?

In any natural disaster like an earthquake, there is always a potential for large casualties, particularly in emerging countries. In this devastating natural disaster in Haiti, approximately 3 million people were affected. This earthquake was the most devastating natural disaster ever experienced in Haiti, the Western Hemisphere's poorest country. It is estimated that 250,000 lives were lost, and 300,000 people were injured. When people are scattered in a large geographical area with no easy transportation or communication options, it is important to reach the affected people as soon as possible to limit the number of people dying from thirst, hunger, and starvation. And the biggest challenge in such situations is to locate people in a large geographical area.

Hence our study here of being able to recognize blue tarps effectively would have made a significant difference in reducing further casualties by enabling rescue works to reach the affected people quickly and provide them with much-needed resources.

f) suitable for one class of predication methods

Do these data seem particularly well-suited to one class of prediction methods, and if so, why?

In this case, the dataset's response variable is a multi-class discrete variable, which we used to create a two-class response variable called class1. The data set in imbalanced and has a large number of records. For this purpose, flexible models tend to perform better than non-flexible models such as LR, LDA.
In our study, KNN was the performing model; even though QDA is more flexible, the LDA and QDA it did not perform as well as LR with the best threshold. This could be due to the underlying imbalance in the dataset.