Disaster-Relief-Project-Part-1

Hai Liu

2025-02-16

Load packages

```
library(tidyverse)
library(tidymodels)
library(discrim) # for LDA and QDA
library(ggcorrplot)  # for correlation plot
library(GGally)  # scatterplot matrix
library(patchwork)  # for combining plots
library(probably)  # for threshold_perf
```

Setup parallel processing

We start a cluster for faster calculations.

```
library(doParallel)
cores<-detectCores()
cl <- makeCluster(cores[1]-1)
registerDoParallel(cl)
```

Load the data and do some data processing and EDA.

```
haiti <- read_csv("HaitiPixels.csv")
```

```
head(haiti)
```

```
## # A tibble: 6 x 4
##   Class        Red Green  Blue
##   <chr>      <dbl> <dbl> <dbl>
## 1 Vegetation    64    67    50
## 2 Vegetation    64    67    50
## 3 Vegetation    64    66    49
## 4 Vegetation    75    82    53
## 5 Vegetation    74    82    54
## 6 Vegetation    72    76    52
```

```
unique(haiti$Class)
```

```
## [1] "Vegetation"       "Soil"             "Rooftop"          "Various Non-Tarp"
## [5] "Blue Tarp"
```

Since we are only interested in the level of "Blue Tarp", I create a new variable BT with only two classes, i.e., "TRUE" for "Blue Tarp" and "FALSE" for everything else.

```
haiti <- haiti |>
  mutate(
    BT = ifelse(Class == "Blue Tarp", "TRUE", "FALSE"),
    BT = factor(BT, levels=c("TRUE", "FALSE"))
  )
```

Have a look at the distributioin of the two classes for the outcome named "BT" (for BlueTarp).

```
haiti |>
    ggplot(aes(x=BT, fill=BT)) +
    geom_bar(position="dodge")
```

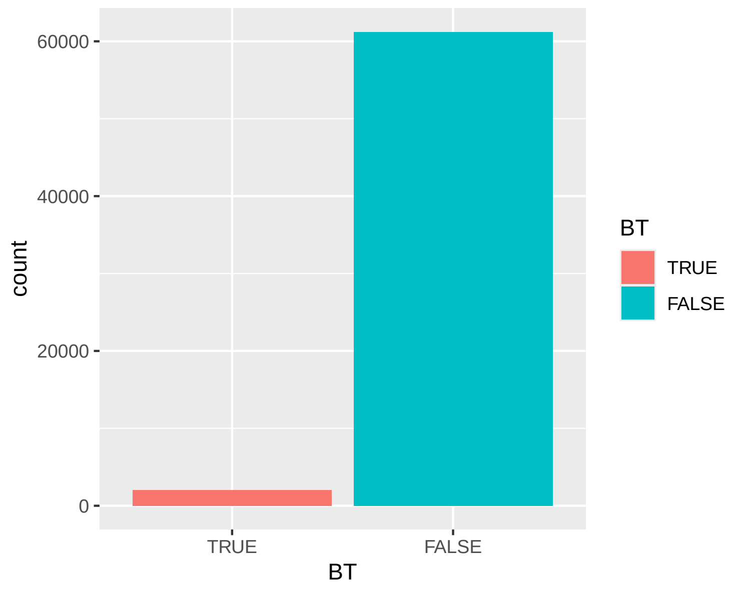


Figure 1: Distribution of Blue Tarp among all the observations.

I can see that the two outcome classes are extremely unbalanced. I will keep this in mind and deal with it later.

Build three classification models, *i.e.*, LDA, QDA, and logistic regression, with cross-validation.

Prepare model workflows

Define the preprocessing steps. In this case, we normalize all numeric predictors.

```
formula <- BT ~ Red + Green + Blue

recipe <- recipe(formula, data=haiti) %>%
    step_normalize(all_numeric_predictors())
```

Specify the three models.

```
logreg_spec <- logistic_reg(mode="classification", engine="glm")
lda_spec <- discrim_linear(mode="classification", engine="MASS")
qda_spec <- discrim_quad(mode="classification", engine="MASS")
```

Combine preprocessing steps and model specification in workflow.

```
logreg_wf <- workflow() %>%
    add_recipe(recipe) %>%
    add_model(logreg_spec)

lda_wf <- workflow() %>%
    add_recipe(recipe) %>%
    add_model(lda_spec)

qda_wf <- workflow() %>%
    add_recipe(recipe) %>%
    add_model(qda_spec)
```

Cross-validation

Define cross-validation approach - 10-fold cross-validation using stratified sampling - Measure performance using ROC-AUC (we also collect accuracy) - Save resample predictions, so that we can build ROC curves using cross-validation results

```
set.seed(6030)
resamples <- vfold_cv(haiti, v=10, strata=BT)
custom_metrics <- metric_set(accuracy, roc_auc, precision, f_meas)
cv_control <- control_resamples(save_pred=TRUE)
```

Cross-validation

```
logreg_cv <- fit_resamples(logreg_wf, resamples, metrics=custom_metrics, control=cv_control)
lda_cv <- fit_resamples(lda_wf, resamples, metrics=custom_metrics, control=cv_control)
qda_cv <- fit_resamples(qda_wf, resamples, metrics=custom_metrics, control=cv_control)
```

Model performance before threshold selection

The performance metrics estimated using 10-fold cross-validation.

```
cv_metrics <- bind_rows(
    collect_metrics(logreg_cv) %>%
        mutate(model="Logistic regression"),
    collect_metrics(lda_cv) %>%
        mutate(model="LDA"),
    collect_metrics(qda_cv) %>%
        mutate(model="QDA")
)

cv_metrics %>%
    select(model, .metric, mean) %>%
    pivot_wider(names_from=".metric", values_from="mean") %>%
    knitr::kable(caption="Cross-validation performance metrics", digits=3)
```

Table 1: Cross-validation performance metrics

model	accuracy	f_meas	precision	roc_auc
Logistic regression	0.995	0.923	0.964	0.998
LDA	0.984	0.761	0.725	0.989
QDA	0.995	0.908	0.989	0.998

Visualization of the same data

```
ggplot(cv_metrics, aes(x=mean, y=model, xmin=mean - std_err, xmax=mean + std_err)) +
    geom_point() +
    geom_linerange() +
    facet_wrap(~ .metric)
```

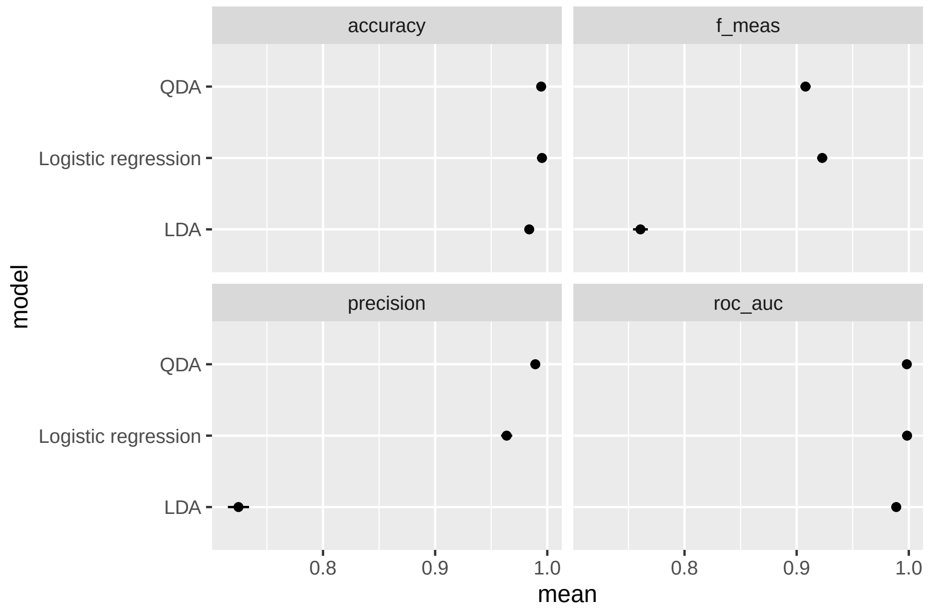


Figure 2: Cross-validation performance metrics

Cross-validation ROC curves

```r
bind_rows(
    collect_predictions(logreg_cv) %>% mutate(model="Logistic regression"),
    collect_predictions(lda_cv) %>% mutate(model="LDA"),
    collect_predictions(qda_cv) %>% mutate(model="QDA")
) %>%
    group_by(model) %>%
    roc_curve(truth=BT, .pred_TRUE, event_level="first") %>%
    autoplot()
```

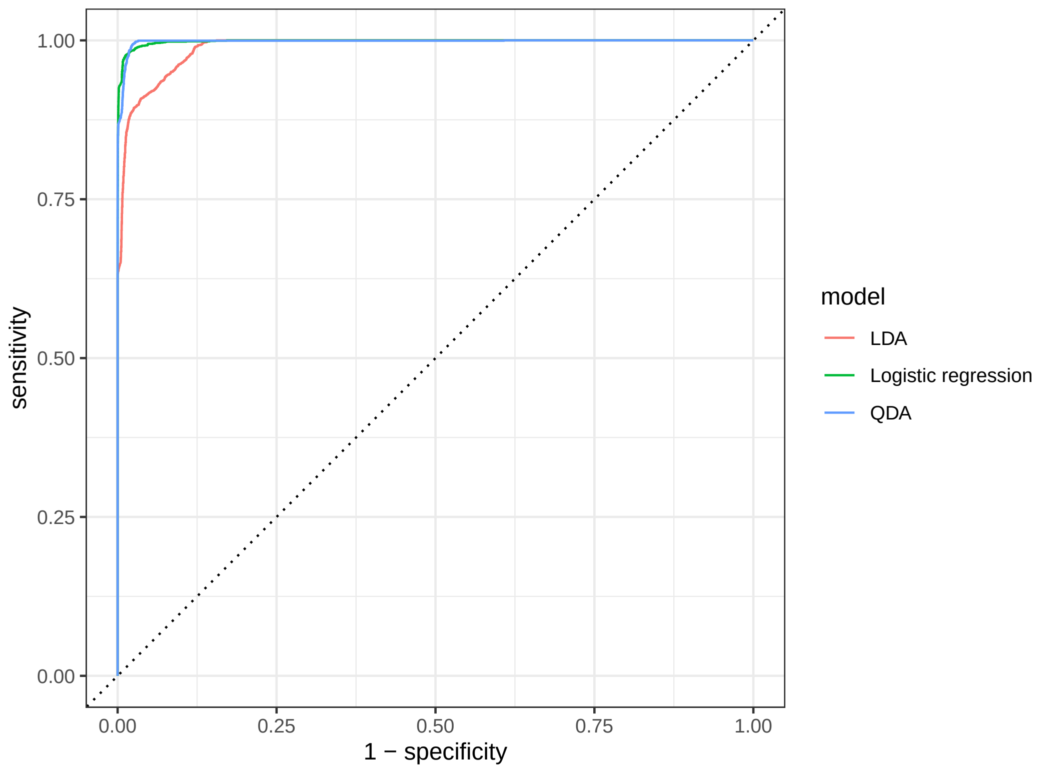


Figure 3: Overlay of cross-validation ROC curves

Threshold selection/Optimization

It is clear that our outcome classes are heavily imbalanced, so we need to adjust the threshold to improve its predictive accuracy and precision. Use package `probably` to explore the threshold. We define two functions to look at the effect of threshold selection on performance metrics and the associated confusion matrices:

```r
threshold_graph <- function(model_cv, model_name) {
    performance <- probably::threshold_perf(collect_predictions(model_cv), BT, .pred_TRUE,
        thresholds=seq(0.05, 0.95, 0.01), event_level="first",
        metrics=metric_set(accuracy, precision, f_meas, kap))
    max_metrics <- performance %>%
        drop_na() %>%
        group_by(.metric) %>%
        filter(.estimate == max(.estimate))
    g <- ggplot(performance, aes(x=.threshold, y=.estimate, color=.metric)) +
        geom_line() +
        geom_point(data=max_metrics, color="black") +
        labs(title=model_name, x="Threshold", y="Metric value") +
        coord_cartesian(ylim=c(0.5, 1.0))
    thresholds <- max_metrics %>%
        select(.metric, .threshold) %>%
        deframe()
    return(list(graph=g, thresholds=thresholds))
}

visualize_conf_mat <- function(model_cv, thresholds, metric) {
    threshold <- thresholds[metric]
    cm <- collect_predictions(model_cv) %>%
        mutate(
            .pred_class = make_two_class_pred(.pred_TRUE, c("TRUE", "FALSE"), threshold=threshold),
        ) %>%
        conf_mat(truth=BT, estimate=.pred_class)
    autoplot(cm, type="heatmap") +
        labs(title=sprintf("Threshold %.2f (%s)", threshold, metric))
}

overview_model <- function(model_cv, model_name) {
    tg <- threshold_graph(model_cv, model_name)
    g1 <- visualize_conf_mat(model_cv, tg$thresholds, "accuracy")
    g2 <- visualize_conf_mat(model_cv, tg$thresholds, "f_meas")
    g3 <- visualize_conf_mat(model_cv, tg$thresholds, "precision")
    tg$graph + (g1 / g2 / g3)
}
```

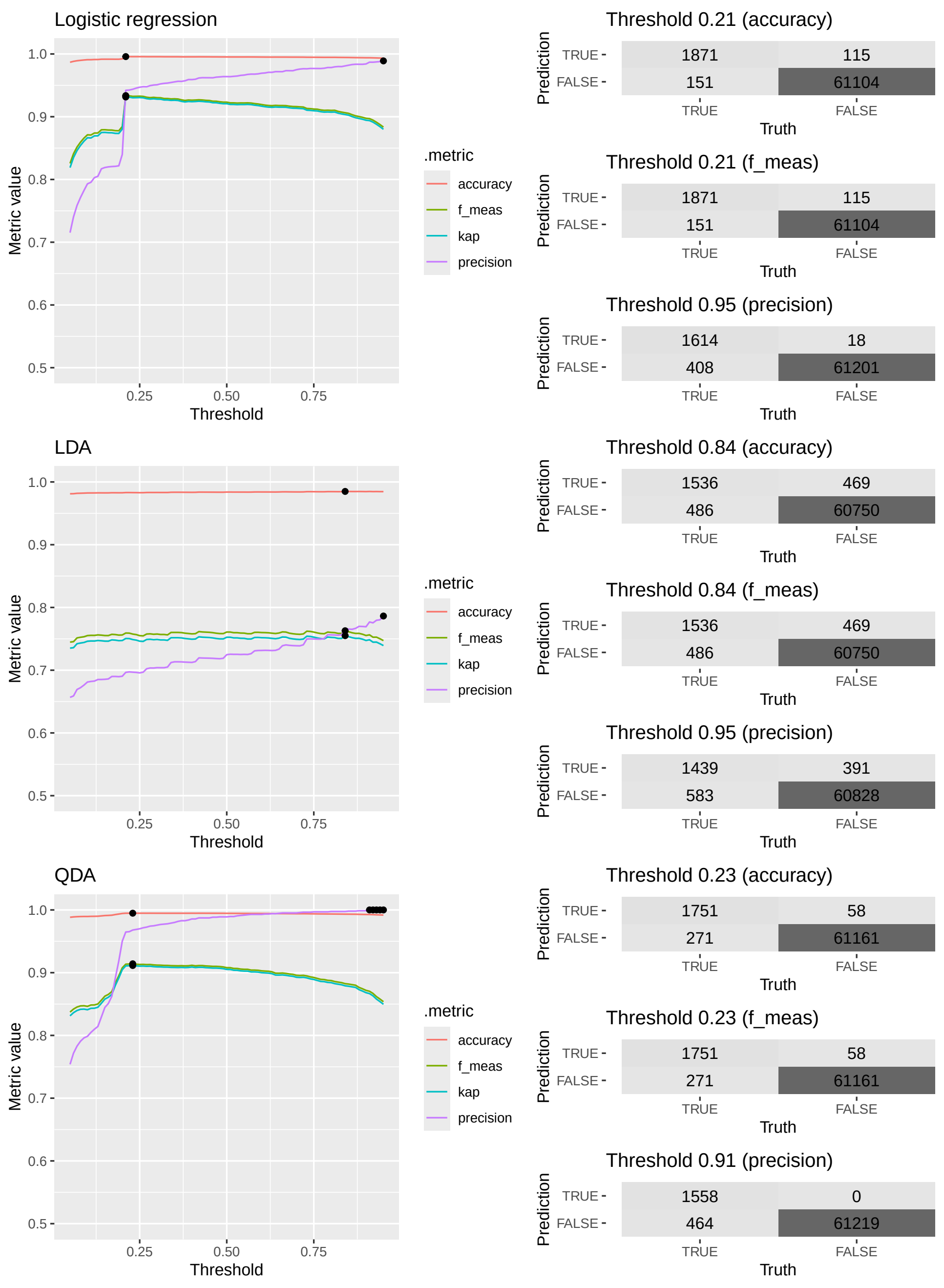
Notes:

- `f_meas` cannot be calculated for high threshold values. In this case, the function `threshold_perf` returns `NA` for the F-measure. We filter out these values using `drop_na()`.

```
g1 <- overview_model(logreg_cv, "Logistic regression")
g2 <- overview_model(lda_cv, "LDA")
g3 <- overview_model(qda_cv, "QDA")

g1 / g2 / g3
```

Since our goal is to identify as many true blue tarps as possible, or predict the most true blue tarps as TRUE, we need to use performance metrics that can balance between precision and recall. Therefore, metrics such as Kappa and F-measure score are supposed to be more appropriate than accuracy. By comparing the confusion matrix among the three models across all the thresholds tested, I find that the logistic regression at a threshold of 0.21 gives the highest F-measure score, which suggests the best balance/trade-off between precision and sensitivity. Also, it is very interesting to see that the Kappa and F-measure are in line with each other all the time in this case for all three models, confirming their usefulness in handling class imbalance.

```
probably::threshold_perf(collect_predictions(logreg_cv), BT, .pred_TRUE,
        thresholds=0.21, event_level="first",
        metrics=metric_set(accuracy, precision, f_meas, kap)) |>
  select(c(.metric, .estimate)) |>
```



Threshold 0.21 (accuracy)

Prediction \ Truth	TRUE	FALSE
TRUE	1871	115
FALSE	151	61104

Threshold 0.21 (f_meas)

Prediction \ Truth	TRUE	FALSE
TRUE	1871	115
FALSE	151	61104

Threshold 0.95 (precision)

Prediction \ Truth	TRUE	FALSE
TRUE	1614	18
FALSE	408	61201

Threshold 0.84 (accuracy)

Prediction \ Truth	TRUE	FALSE
TRUE	1536	469
FALSE	486	60750

Threshold 0.84 (f_meas)

Prediction \ Truth	TRUE	FALSE
TRUE	1536	469
FALSE	486	60750

Threshold 0.95 (precision)

Prediction \ Truth	TRUE	FALSE
TRUE	1439	391
FALSE	583	60828

Threshold 0.23 (accuracy)

Prediction \ Truth	TRUE	FALSE
TRUE	1751	58
FALSE	271	61161

Threshold 0.23 (f_meas)

Prediction \ Truth	TRUE	FALSE
TRUE	1751	58
FALSE	271	61161

Threshold 0.91 (precision)

Prediction \ Truth	TRUE	FALSE
TRUE	1558	0
FALSE	464	61219

Figure 4: Metrics as a function of model performance

```
  pivot_wider(names_from=".metric", values_from=".estimate") |>
  knitr::kable(caption="Performance metrics for the
               logistic regression at threshold of 0.21", digits=3)
```

Table 2: Performance metrics for the logistic regression at threshold of 0.21

accuracy	precision	f_meas	kap
0.996	0.942	0.934	0.931

At this threshold, the logistic regression has an accuracy of 99.6% and precision of 94.2%. In addition, the True Positive Rate (TPR) is: 1871/(151+1871)=92.53%, and the False Positive Rate (FPR) is: 115/(61104+115)=0.19%.

Next, I compared the ROC curves for the logistic regression between the cross-validation predictions and the predictions on the full training data set to see if the logistic regression was overfitting the training data.

```
logreg_full <- fit(logreg_wf, data=haiti)

cv_roc <- collect_predictions(logreg_cv) %>%
    roc_curve(truth=BT, .pred_TRUE, event_level="first")

full_roc <- augment(logreg_full, new_data=haiti) %>%
    roc_curve(truth=BT, .pred_TRUE, event_level="first")

ggplot() +
    geom_path(data=cv_roc, aes(x=1 - specificity, y=sensitivity), color="blue") +
    geom_path(data=full_roc, aes(x=1 - specificity, y=sensitivity), color="red", linetype="dashed") +
    geom_abline(lty=2)
```

As we can see that the ROC curve for the cross-validation predictions is almost identical to the ROC curve for the predictions on the full training set, indicating that the logistic regression model is not overfitting the training data. Therefore, for the training data, the logistic regression model performs the best among the three models and has the best performance metrics at a threshold of 0.21.

Process the holdout set.

```
col_names <- c('ID','X','Y','Map X','Map Y','Lat','Lon','Red','Green','Blue')
```

```
haiti_bt1 <- read_table("./HoldOutData/orthovnir067_ROI_Blue_Tarps.txt", comment=";", col_nam
  select(Red, Green, Blue) %>%
  mutate(BT = "TRUE")

haiti_bt2 <- read_table("./HoldOutData/orthovnir069_ROI_Blue_Tarps.txt", comment=";", col_nam
  select(Red, Green, Blue) %>%
  mutate(BT = "TRUE")

haiti_bt3 <- read_table("./HoldOutData/orthovnir078_ROI_Blue_Tarps.txt", comment=";", col_nam
  select(Red, Green, Blue) %>%
  mutate(BT = "TRUE")
```

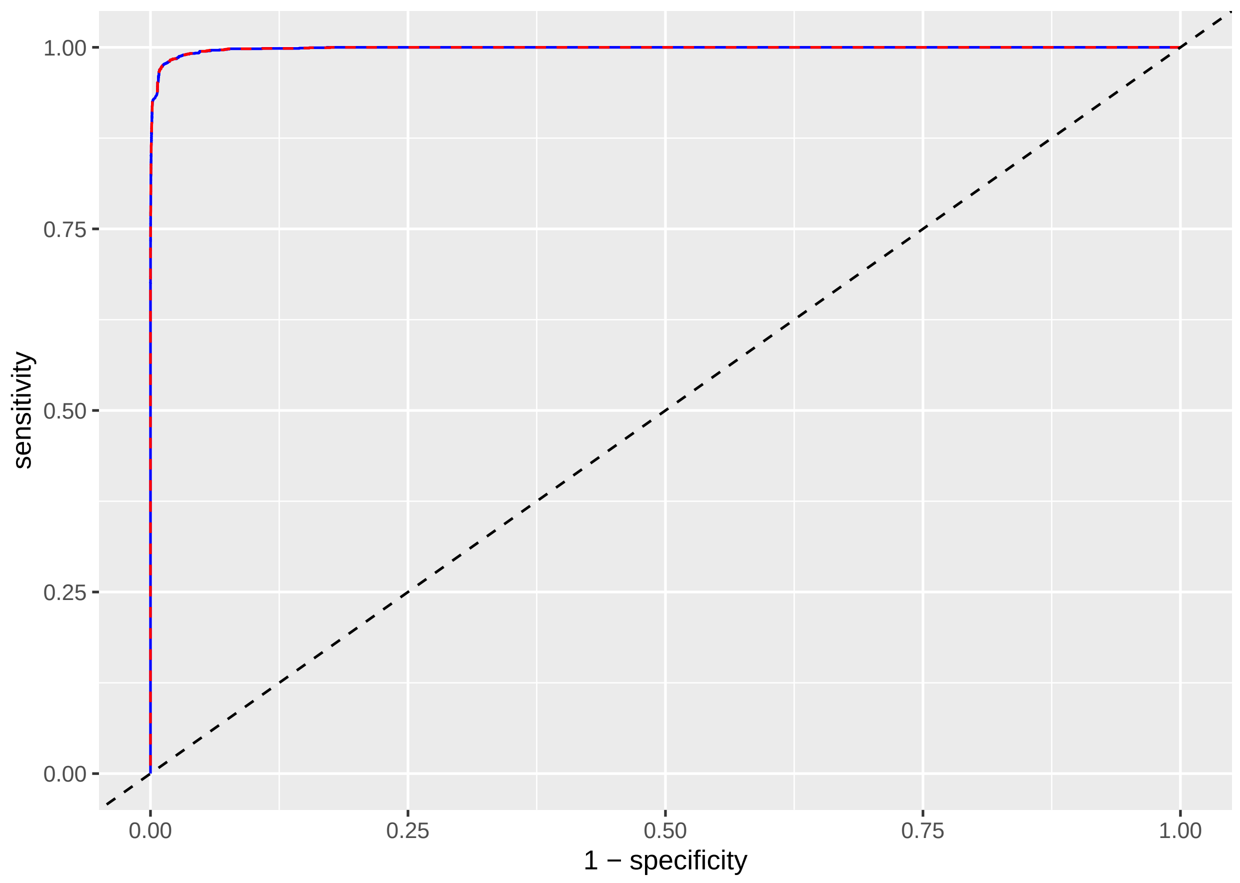
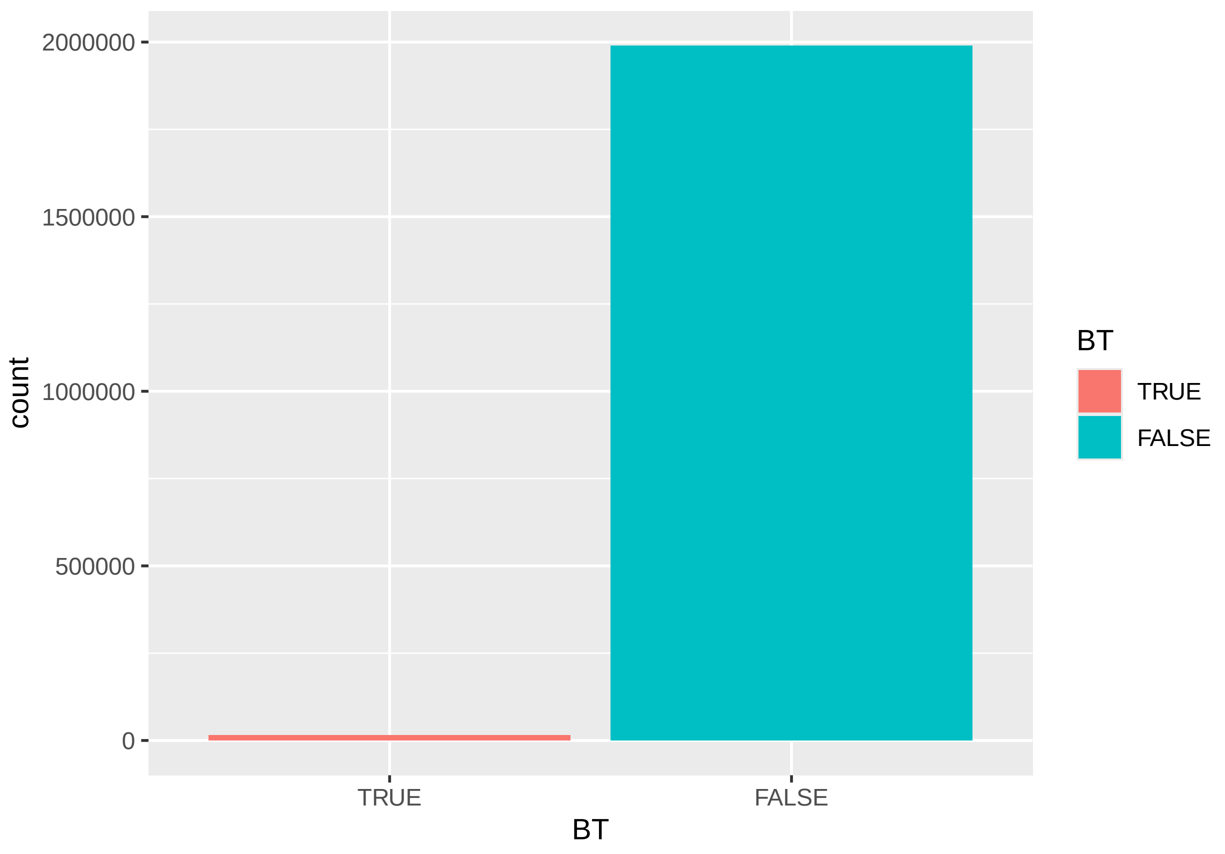


Figure 5: ROC curve comarison between corss-valication and full data set predictions

```
haiti_nbt1 <- read_table("./HoldOutData/orthovnir057_ROI_NON_Blue_Tarps.txt", comment=";", col_names=col
  select(Red, Green, Blue) %>%
  mutate(BT = "FALSE")

haiti_nbt2 <- read_table("./HoldOutData/orthovnir067_ROI_NOT_Blue_Tarps.txt", comment=";", col_names=col
  select(Red, Green, Blue) %>%
  mutate(BT = "FALSE")

haiti_nbt3 <- read_table("./HoldOutData/orthovnir069_ROI_NOT_Blue_Tarps.txt", comment=";", col_names=col
  select(Red, Green, Blue) %>%
  mutate(BT = "FALSE")

haiti_nbt4 <- read_table("./HoldOutData/orthovnir078_ROI_NON_Blue_Tarps.txt", comment=";", col_names=col
  select(Red, Green, Blue) %>%
  mutate(BT = "FALSE")
```

```
haiti_holdout <- bind_rows(haiti_bt1, haiti_bt2, haiti_bt3,
                           haiti_nbt1, haiti_nbt2, haiti_nbt3, haiti_nbt4) %>%
  mutate(BT = factor(BT, levels=c("TRUE", "FALSE")))

haiti_holdout %>%
  ggplot(aes(x=BT, fill=BT)) +
  geom_bar(position="dodge")
```



It is very noticeable that the two classes of the outcome are even more imbalanced than that in the training data.

The performance of the three models on the holdout set at respective optimal threshold.

```r
predict_at_threshold <- function(model, data, threshold) {
  return(model %>%
           augment(data) %>%
           mutate(.pred_class = make_two_class_pred(.pred_TRUE,c("TRUE", "FALSE"), threshold=threshold)
}

calculate_metrics_at_threshold <- function(model, train, holdout, model_name, threshold) {
    bind_rows(
        # Accuracy of training set
        bind_cols(
            model=model_name, dataset="train", threshold=threshold,
            metrics(predict_at_threshold(model, train, threshold), truth=BT, estimate=.pred_class),
        ),
        # AUC of ROC curve of training set
        bind_cols(
            model=model_name, dataset="train", threshold=threshold,
            roc_auc(model %>% augment(train), BT, .pred_TRUE, event_level="first"),
        ),
        # Accuracy of holdout set
        bind_cols(
            model=model_name, dataset="holdout", threshold=threshold,
            metrics(predict_at_threshold(model, holdout, threshold), truth=BT, estimate=.pred_class),
        ),
        # AUC of ROC curve of holdout set
        bind_cols(
            model=model_name, dataset="holdout", threshold=threshold,
            roc_auc(model %>% augment(holdout), BT, .pred_TRUE, event_level="first"),
        ),
    )
}
```

```r
lda_full <- fit(lda_wf, data=haiti)
qda_full <- fit(qda_wf, data=haiti)

logreg_opti_th <- 0.21
lda_opti_th <- 0.84
qda_opti_th <- 0.23
```

```r
logreg_metrics <- calculate_metrics_at_threshold(logreg_full, haiti, haiti_holdout, "Logistic regressio
```

```r
lda_metrics <- calculate_metrics_at_threshold(lda_full, haiti, haiti_holdout, "LDA", lda_opti_th)
```

```r
qda_metrics <- calculate_metrics_at_threshold(qda_full, haiti, haiti_holdout, "QDA", qda_opti_th)
```

```r
metrics_table <- function(all_metrics, caption) {
  all_metrics %>%
    pivot_wider(names_from=.metric, values_from=.estimate) %>%
    select(-.estimator) %>%
    knitr::kable(caption=caption, digits=3)
}

metrics_at_threshold <- bind_rows(
    logreg_metrics,
    lda_metrics,
    qda_metrics,
) %>% arrange(dataset)

metrics_table(metrics_at_threshold, "Performance metrics with optimized threshold")
```

Table 3: Performance metrics with optimized threshold

model	dataset	threshold	accuracy	kap	roc_auc
Logistic regression	holdout	0.21	0.965	0.279	0.999
LDA	holdout	0.84	0.984	0.394	0.992
QDA	holdout	0.23	0.995	0.683	0.992
Logistic regression	train	0.21	0.996	0.932	0.999
LDA	train	0.84	0.985	0.755	0.989
QDA	train	0.23	0.995	0.911	0.998

It is interesting that while the logistic regression at its optimal threshold performs the best on the training data, the QDA at its optimal threshold performs the best on the holdout dataset. Nevertheless, the roc_auc value still suggests that the logistic regression is still the best model although the optimal threshold derived from the training data performs very poorly on the test/holdout data. This may be due to the further imbalance of the two classes of the outcome in the holdout data.

Stop cluster

```
stopCluster(cl)
registerDoSEQ()
```