DS6030 Project Data Evaluation

Tyler Hobbs

2025-06-17

```
library(tidyverse)
library(GGally)
library(tidymodels)
library(discrim)
library(patchwork)
library(probably)
library(vip)
```

Loading the data and Pre-processing

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

```
## Rows: 63241 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): Class
## dbl (3): Red, Green, Blue
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
test1<-read_table("orthovnir057_ROI_NON_Blue_Tarps.txt",show_col_types = FALSE)
test1_sub<-test1[1:10]
colnames(test1_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")
test2<-read_table("orthovnir067_ROI_Blue_Tarps_data.txt",show_col_types = FALSE)
test3<-read_table("orthovnir067_ROI_Blue_Tarps.txt",show_col_types = FALSE)
test3_sub<-test3[1:10]
colnames(test3_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")
test4<-read_table("orthovnir067_ROI_NOT_Blue_Tarps.txt",show_col_types = FALSE)
test4_sub<-test4[1:10]
colnames(test4_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")
test5<-read_table("orthovnir069_ROI_Blue_Tarps.txt",show_col_types = FALSE)
test5_sub<-test5[1:10]
colnames(test5_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")
test6<-read_table("orthovnir069_ROI_NOT_Blue_Tarps.txt",show_col_types = FALSE)
test6_sub<-test6[1:10]
colnames(test6_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")
test7<-read_table("orthovnir078_ROI_Blue_Tarps.txt",show_col_types = FALSE)
test7_sub<-test7[1:10]
colnames(test7_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")
test8<-read_table("orthovnir078_ROI_NON_Blue_Tarps.txt",show_col_types = FALSE)
test8_sub<-test8[1:10]
colnames(test8_sub)<-c("ID","X","Y","Map X","Map Y", "Lat","Lon","B1","B2","B3")

combined_df_holdout<-bind_rows(test1_sub,test3_sub,test4_sub,test5_sub,test6_sub,test7_s
ub,test8_sub)
df_fig<-combined_df_holdout[8:10]
df_ID<-combined_df_holdout[1]
df_figID<-cbind(df_ID,df_fig)
lonlatcords<-combined_df_holdout[6:10]
trainwoclass<-train[2:4]
```

Pre-processing

```
NonBlueholdout<-bind_rows(test1_sub,test4_sub,test6_sub,test7_sub,test8_sub)
NonBlueholdout_new<-NonBlueholdout[8:10]
names(NonBlueholdout_new)[1]<-"Red"
names(NonBlueholdout_new)[2]<-"Green"
names(NonBlueholdout_new)[3]<-"Blue"
NonBlueholdout_new <- NonBlueholdout_new %>%
  mutate(Class=1)
NonBlueholdout_new <- NonBlueholdout_new %>%
mutate(Class=factor(Class,labels=c("1")))%>%
  mutate(Class=case_when(
    Class == 1 ~ "Non-Blue_Tarp"
  ))
Blueholdout<-bind_rows(test2,test3_sub,test5_sub)
Blueholdout_new<-Blueholdout[1:3]
names(Blueholdout_new)[1]<-"Red"
names(Blueholdout_new)[2]<-"Green"
names(Blueholdout_new)[3]<-"Blue"
Blueholdout_new <- Blueholdout_new %>%
  mutate(Class=1)
Blueholdout_new <- Blueholdout_new %>%
mutate(Class=factor(Class,labels=c("1")))%>%
  mutate(Class=case_when(
    Class == 1 ~ "Blue_Tarp"
  ))
holdout<-bind_rows(Blueholdout_new,NonBlueholdout_new)
```

```
holdout<-holdout %>%
  mutate(type=case_when(
    Class == "Non-Blue_Tarp" ~ 0,
    Class == "Blue_Tarp" ~ 1
    ))
```

```
holdout<-holdout %>%
    mutate(
        group = paste(type, Class, sep="_"),
        group = factor(group),
    )
```

Training

```
Haiti_train<-train %>%
  mutate(type=case_when(
    Class == "Rooftop" ~ "Non-Blue_Tarp",
    Class == "Soil" ~ "Non-Blue_Tarp",
    Class == "Various Non-Tarp" ~ "Non-Blue_Tarp",
    Class == "Vegetation" ~ "Non-Blue_Tarp",
    Class == "Blue Tarp" ~ "Blue_Tarp"
  ))%>%
  mutate(Class=case_when(
    Class == "Rooftop" ~ "Non-Blue_Tarp",
    Class == "Soil" ~ "Non-Blue_Tarp",
    Class == "Various Non-Tarp" ~ "Non-Blue_Tarp",
    Class == "Vegetation" ~ "Non-Blue_Tarp",
    Class == "Blue Tarp" ~ "Blue_Tarp"
  ))%>%
    mutate(
        type=factor(type, levels=c("Blue_Tarp", "Non-Blue_Tarp")))%>%
    mutate(
        group = paste(type, Class, sep="_"),
        group = factor(group),
    )
set.seed(1)
formula<-Class~Red + Green + Blue
Haiti_recipe <- recipe(formula, data=Haiti_train) %>%
    step_normalize(all_numeric_predictors())
names(df_fig)[1]<-"Red"
names(df_fig)[2]<-"Green"
names(df_fig)[3]<-"Blue"
```

Logistic, LDA, and QDA Models

```
holdout<-holdout%>%
  mutate(
    Class=factor(Class))
Haiti_train<-Haiti_train%>%
  mutate(
    Class=factor(Class))
logreg_model_full <- logistic_reg(mode="classification", engine="glm") %>%
  fit(formula, Haiti_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
lda_model <- discrim_linear(mode="classification", engine="MASS") %>%
  fit(formula, Haiti_train)
qda_model <- discrim_quad(mode="classification", engine="MASS") %>%
  fit(formula,Haiti_train)
```

```
resamples <- vfold_cv(holdout, v=10, strata=type)
custom_metrics <- metric_set(roc_auc, accuracy)
cv_control <- control_resamples(save_pred=TRUE)
```

```
calculate_metrics <- function(model, train, test, model_name) {
bind_rows(
# Accuracy of training set
bind_cols(
model=model_name,
dataset="train",
metrics(model %>% augment(Haiti_train), truth=Class, estimate=.pred_class),
),
# AUC of ROC curve of training set
bind_cols(
model=model_name,
dataset="train",
roc_auc(model %>% augment(Haiti_train), Class, .pred_Blue_Tarp, event_level="first"),
),
# Accuracy of holdout set
bind_cols(
model=model_name,
dataset="test",
metrics(model %>% augment(holdout), truth=Class, estimate=.pred_class),
),
# AUC of ROC curve of holdout set
bind_cols(
model=model_name,
dataset="test",
roc_auc(model %>% augment(holdout), Class, .pred_Blue_Tarp, event_level="first"),
),
)
}
```

```
metrics_table <- function(all_metrics, caption) {
all_metrics <- all_metrics %>% arrange(model, desc(dataset))
all_metrics %>%
pivot_wider(names_from=.metric, values_from=.estimate) %>%
dplyr::select(-.estimator) %>%
knitr::kable(caption=caption, digits=5) %>%
kableExtra::kable_styling(full_width=FALSE)
}
```

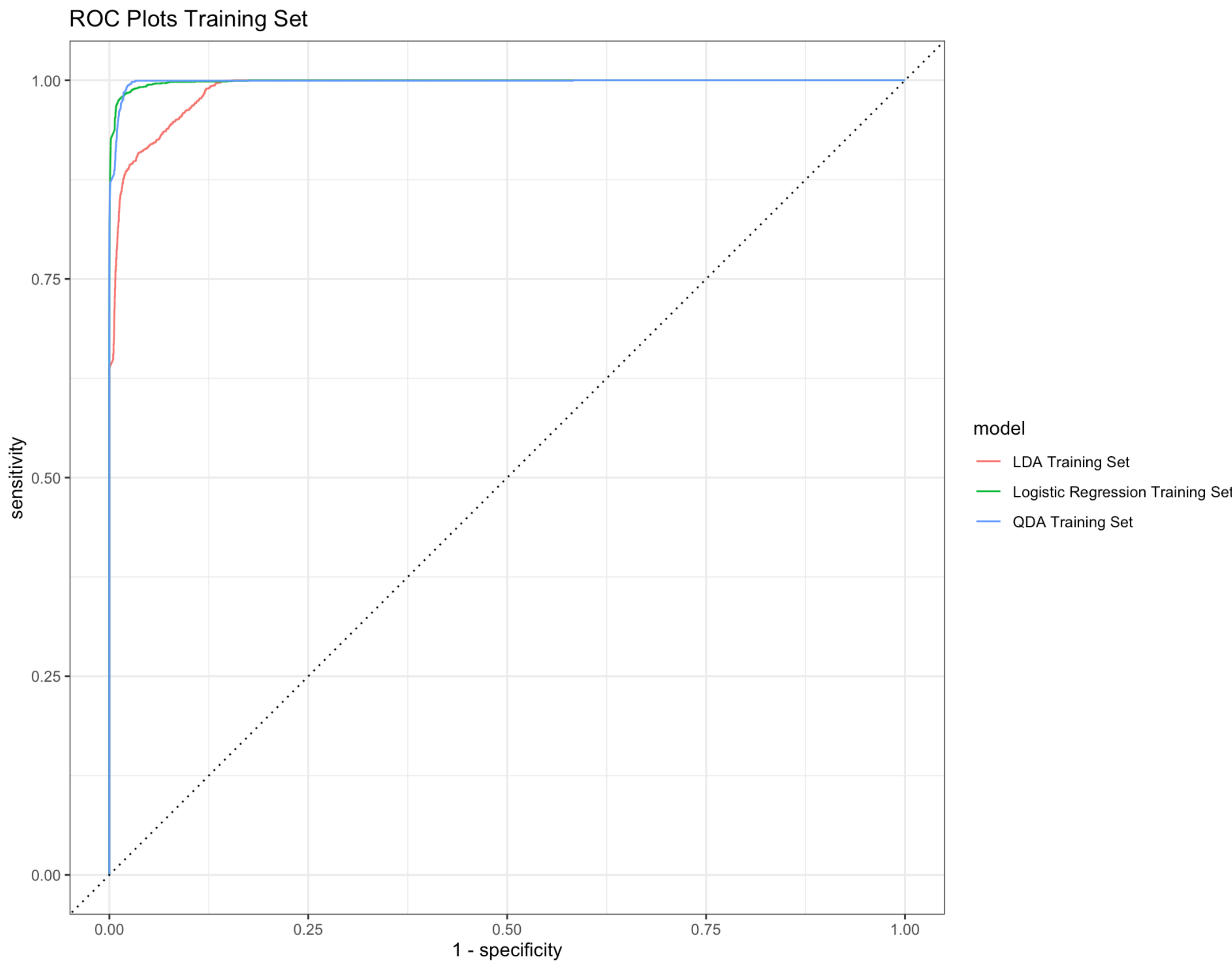
```
all_metrics <- bind_rows(
calculate_metrics(logreg_model_full, Haiti_train, holdout, "Logistic Regression"),
calculate_metrics(lda_model, Haiti_train, holdout, "LDA"),
calculate_metrics(qda_model, Haiti_train, holdout, "QDA"),
)
metrics_table(all_metrics, "Metrics for the classification models")
```

Metrics for the classification models

model	dataset	accuracy	kap	roc_auc
LDA	train	0.98397	0.75336	0.98888
LDA	test	0.98014	0.38800	0.99146
Logistic Regression	train	0.99529	0.92073	0.99851
Logistic Regression	test	0.98817	0.56108	0.99840
QDA	train	0.99461	0.90604	0.99822
QDA	test	0.99491	0.68267	0.99209

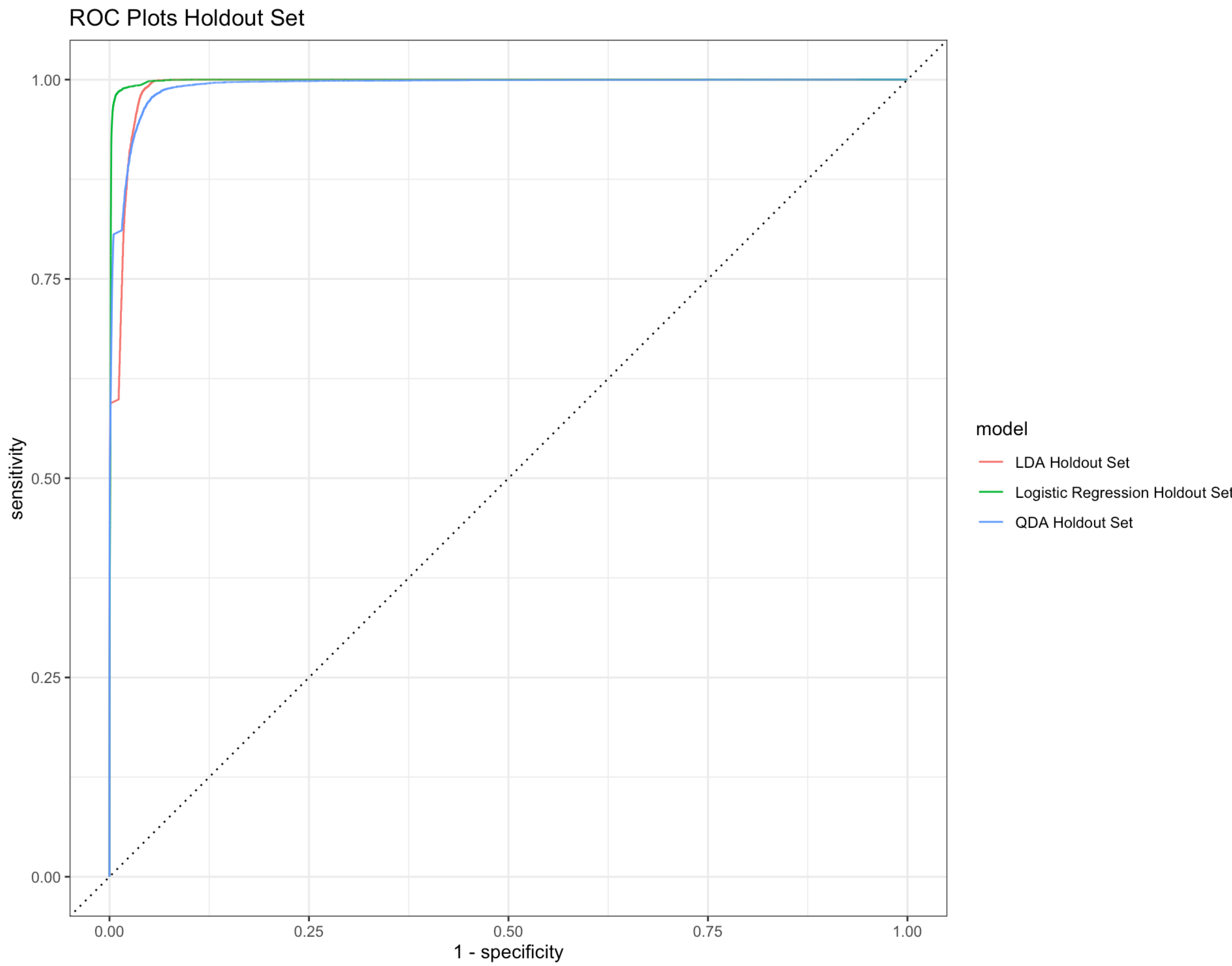
```
get_roc_plot <- function(model, data, model_name) {
roc_data <- model %>%
augment(Haiti_train) %>%
roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first")
roc_auc(model %>% augment(Haiti_train), Class, .pred_Blue_Tarp, event_level="first")
g <- autoplot(roc_data) +
labs(title=model_name)
return(g)
}
```

```
bind_rows(
augment(logreg_model_full, Haiti_train) %>% mutate(model="Logistic Regression Training S
et"),
augment(lda_model, Haiti_train) %>% mutate(model="LDA Training Set"),
augment(qda_model, Haiti_train) %>% mutate(model="QDA Training Set")
) %>%
group_by(model) %>%
roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first") %>%
autoplot() + labs(title="ROC Plots Training Set")
```

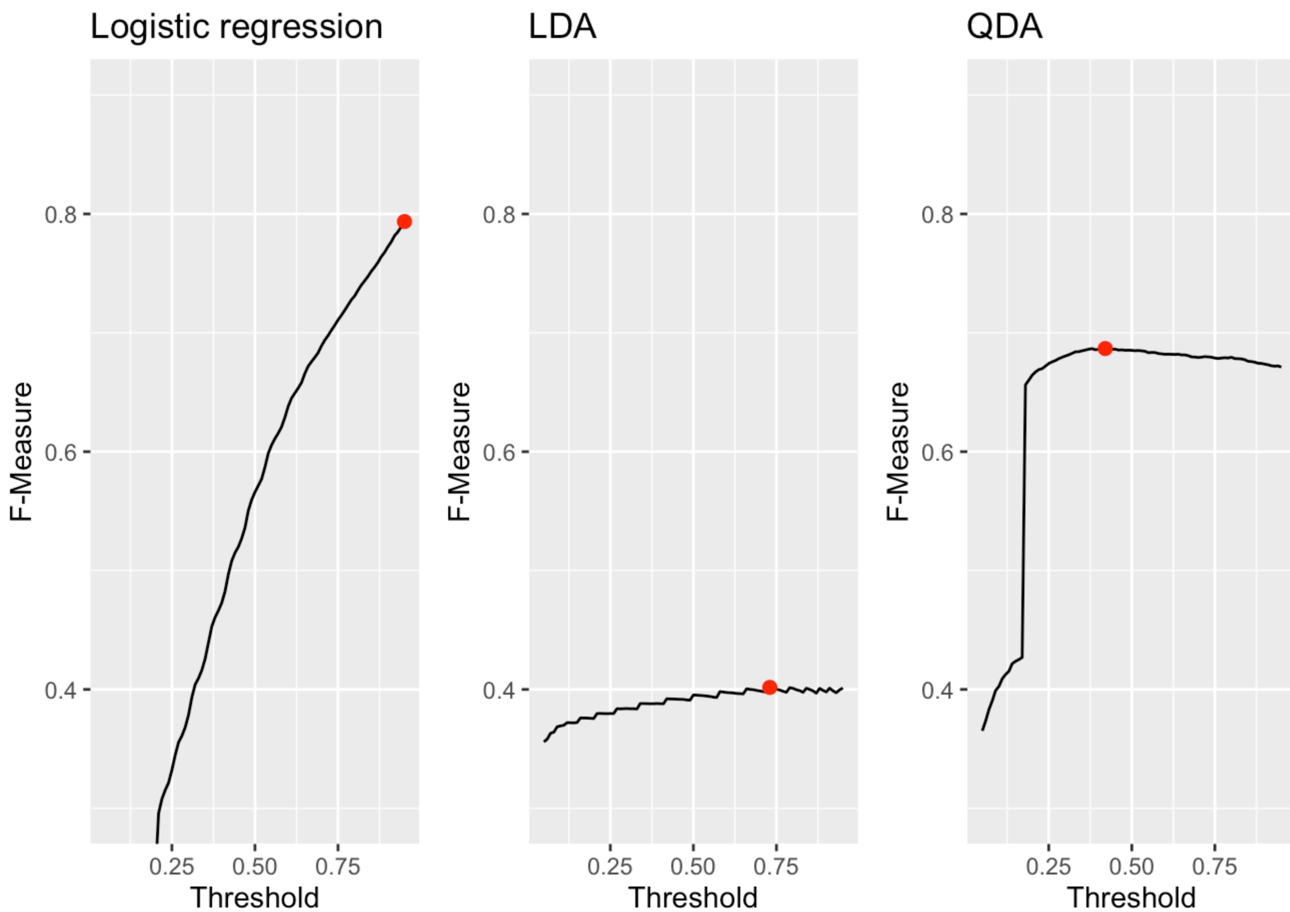



```
get_roc_plot <- function(model, data, model_name) {
roc_data <- model %>%
augment(holdout) %>%
roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first")
roc_auc(model %>% augment(holdout), Class, .pred_Blue_Tarp, event_level="first")
g <- autoplot(roc_data) +
labs(title=model_name)
return(g)
}
```

```
bind_rows(
augment(logreg_model_full, holdout) %>% mutate(model="Logistic Regression Holdout Set"),
augment(lda_model, holdout) %>% mutate(model="LDA Holdout Set"),
augment(qda_model, holdout) %>% mutate(model="QDA Holdout Set")
) %>%
group_by(model) %>%
roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first") %>%
autoplot() + labs(title="ROC Plots Holdout Set")
```



```
threshold_scan <- function(model, data, model_name) {
threshold_data <- model %>%
augment(holdout) %>%
probably::threshold_perf(Class, .pred_Blue_Tarp,
thresholds=seq(0.05, 0.95, 0.01), event_level="first",
metrics=metric_set(f_meas))
opt_threshold <- threshold_data %>%
arrange(-.estimate) %>%
first()
  thresh <- ggplot(threshold_data, aes(x=.threshold, y=.estimate)) +
  geom_line() +
  geom_point(data=opt_threshold, color="red", size=2) +
  labs(title=model_name, x="Threshold", y="F-Measure") +
  coord_cartesian(ylim=c(0.3, 0.9))
  return(list(
  graph=thresh,
  threshold=opt_threshold %>%
  pull(.threshold)
  ))
}
thresh1 <- threshold_scan(logreg_model_full, test, "Logistic regression")
thresh2 <- threshold_scan(lda_model, test, "LDA")
thresh3 <- threshold_scan(qda_model, test, "QDA")
logreg_threshold <- thresh1$threshold
lda_threshold <- thresh2$threshold
qda_threshold <- thresh3$threshold
thresh1$graph + thresh2$graph + thresh3$graph
```




```
thresh1$threshold
```

```
## [1] 0.95
```

```
thresh2$threshold
```

```
## [1] 0.73
```

```
thresh3$threshold
```

```
## [1] 0.42
```

```
logthresh<-0.95
ldathresh<-0.73
qdathresh<-0.42
```

```
predict_at_threshold <- function(model, data, threshold) {
return(
model %>%
augment(holdout) %>%
mutate(.pred_class = make_two_class_pred(.pred_Blue_Tarp,
c("Blue_Tarp", "Non-Blue_Tarp"), threshold=threshold)
)
)
}
predictions_logreg <- predict_at_threshold(logreg_model_full, holdout, logreg_threshold)
predictions_qda <- predict_at_threshold(qda_model, holdout, lda_threshold)
predictions_qda_05 <- predict_at_threshold(qda_model, holdout, qda_threshold)

conf_mat(predictions_logreg, truth=Class, estimate=.pred_class)
```

```
##                Truth
## Prediction      Blue_Tarp Non-Blue_Tarp
##   Blue_Tarp         14991          7069
##   Non-Blue_Tarp       729       1985834
```

```
conf_mat(predictions_qda, truth=Class, estimate=.pred_class)
```

```
##                Truth
## Prediction      Blue_Tarp Non-Blue_Tarp
##   Blue_Tarp         10361          4405
##   Non-Blue_Tarp      5359       1988498
```

```
conf_mat(predictions_qda_05, truth=Class, estimate=.pred_class)
```

```
##                Truth
## Prediction      Blue_Tarp Non-Blue_Tarp
##   Blue_Tarp         11444          6164
##   Non-Blue_Tarp      4276       1986739
```

```
metrics<-metric_set(yardstick::accuracy,yardstick::sens,yardstick::spec,yardstick::f_mea
s,yardstick::j_index)
```

```
calculate_metrics_at_threshold <- function(model, train, test, model_name, threshold) {
    bind_rows(
        bind_cols(
            model=model_name, dataset="train", threshold=threshold,
            metrics(predict_at_threshold(model, train, threshold),
                truth=Class, estimate=.pred_class),
        ),
        bind_cols(
            model=model_name, dataset="test", threshold=threshold,
            metrics(predict_at_threshold(model, holdout, threshold),
                truth=Class, estimate=.pred_class),
        ),

    )
}

metrics_at_threshold <- bind_rows(
    calculate_metrics_at_threshold(logreg_model_full, Haiti_train, holdout, "Logistic re
gression", logthresh),
    calculate_metrics_at_threshold(lda_model, Haiti_train, holdout, "LDA", ldathresh),
    calculate_metrics_at_threshold(qda_model, Haiti_train, holdout, "QDA", qdathresh),
) %>% arrange(dataset)

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

Performance metrics with optimized threshold

model	dataset	threshold	accuracy	sens	spec	f_meas	j_index
LDA	train	0.73	0.98151	0.79300	0.98300	0.40168	0.77600
LDA	test	0.73	0.98151	0.79300	0.98300	0.40168	0.77600
Logistic regression	train	0.95	0.99612	0.95363	0.99645	0.79359	0.95008
Logistic regression	test	0.95	0.99612	0.95363	0.99645	0.79359	0.95008
QDA	train	0.42	0.99480	0.72799	0.99691	0.68675	0.72490
QDA	test	0.42	0.99480	0.72799	0.99691	0.68675	0.72490

```
Haiti_train<-train %>%
  mutate(type=case_when(
    Class == "Rooftop" ~ "Non-Blue_Tarp",
    Class == "Soil" ~ "Non-Blue_Tarp",
    Class == "Various Non-Tarp" ~ "Non-Blue_Tarp",
    Class == "Vegetation" ~ "Non-Blue_Tarp",
    Class == "Blue Tarp" ~ "Blue_Tarp"
  ))%>%
  mutate(Class=case_when(
    Class == "Rooftop" ~ "Non-Blue_Tarp",
    Class == "Soil" ~ "Non-Blue_Tarp",
    Class == "Various Non-Tarp" ~ "Non-Blue_Tarp",
    Class == "Vegetation" ~ "Non-Blue_Tarp",
    Class == "Blue Tarp" ~ "Blue_Tarp"
  ))%>%
    mutate(
        type=factor(type, levels=c("Blue_Tarp", "Non-Blue_Tarp")))%>%
    mutate(
        group = paste(type, Class, sep="_"),
        group = factor(group),
    )
set.seed(1)
formula<-Class~Red + Green + Blue
Haiti_recipe <- recipe(formula, data=Haiti_train) %>%
    step_normalize(all_numeric_predictors())
tuningtrain<-Haiti_train[1:4]
tuningtrain_sub<- tuningtrain %>%
  mutate(Class=factor(Class,labels=c("Non-Blue_Tarp","Blue_Tarp")))%>%
  mutate(case_when(
    Class == "Blue_Tarp" ~ 1,
    Class == "Non-Blue_Tarp" ~ 0
  ))
```

Random Forests:

```
tuningtrain$Red<-as.numeric(tuningtrain$Red)
tuningtrain$Green<-as.numeric(tuningtrain$Green)
tuningtrain$Blue<-as.numeric(tuningtrain$Blue)
tuningtrain$Class<-as.character(tuningtrain$Class)
formula<-Class~ Red + Green + Blue
rec <- recipe(formula, data=tuningtrain)
class(tuningtrain$Class)
```

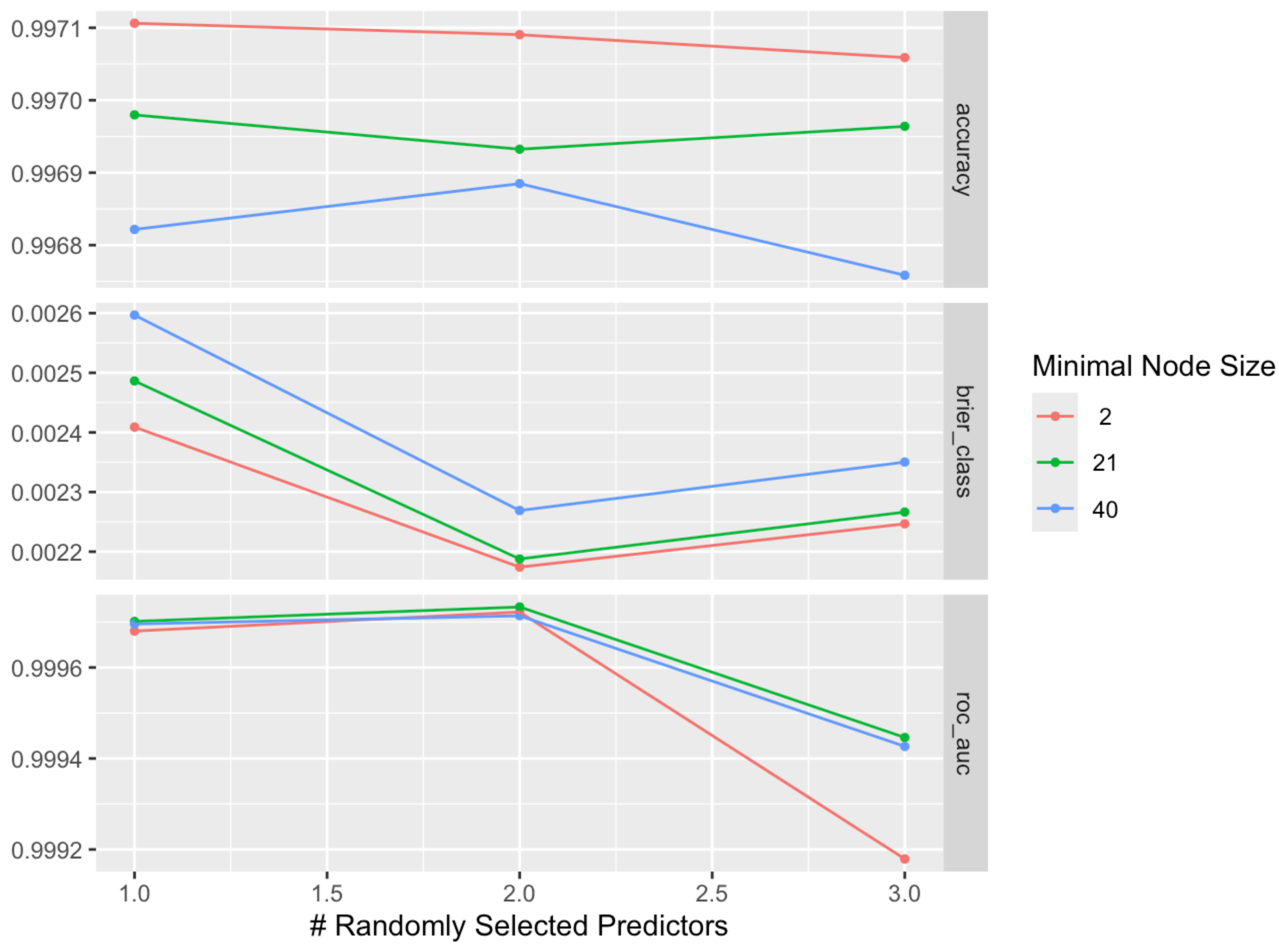
```
## [1] "character"
```

```
forest<-rand_forest(mode="classification", trees=500,min_n = tune(), mtry=tune()) %>%
    set_engine("ranger",importance="impurity")

forests_workflow <- workflow() %>%
    add_recipe(rec) %>%
    add_model(forest)
```

```
parameters <- extract_parameter_set_dials(forests_workflow)%>%
update(
mtry = mtry(c(1, 3)),
min_n = min_n(c(2, 40))
)
```

```
set.seed(1)
tune_results_forest <- tune_grid(forests_workflow,
                          resamples=vfold_cv(tuningtrain),
                          grid=grid_regular(parameters))
```

```
autoplot(tune_results_forest)
```




```
show_best(tune_results_forest,metric="roc_auc")
```

```
## # A tibble: 5 × 8
##    mtry min_n .metric .estimator  mean     n   std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>     <dbl> <chr>
## 1     2    21 roc_auc binary      1.00    10 0.0000348 Preprocessor1_Model5
## 2     2     2 roc_auc binary      1.00    10 0.0000344 Preprocessor1_Model2
## 3     2    40 roc_auc binary      1.00    10 0.0000388 Preprocessor1_Model8
## 4     1    21 roc_auc binary      1.00    10 0.0000401 Preprocessor1_Model4
## 5     1    40 roc_auc binary      1.00    10 0.0000383 Preprocessor1_Model7
```

```
best_parameters <- select_best(tune_results_forest, metric="roc_auc")
best_workflow <- forests_workflow %>%
    finalize_workflow(best_parameters) %>%
    fit(tuningtrain)
best_workflow
```

```
## ══ Workflow [trained] ══════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: rand_forest()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────
## 0 Recipe Steps
##
## ── Model ───────────────────────────────────────────────────────────────────────
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~2L,      x), num.trees = ~500, min.node.size = min_rows(~21L, x),      importance = ~"impurity", num.threads = 1, verbose = FALSE,      seed = sample.int(10^5, 1), probability = TRUE)
##
## Type:                             Probability estimation
## Number of trees:                  500
## Sample size:                      63241
## Number of independent variables:  3
## Mtry:                             2
## Target node size:                 21
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.002167524
```

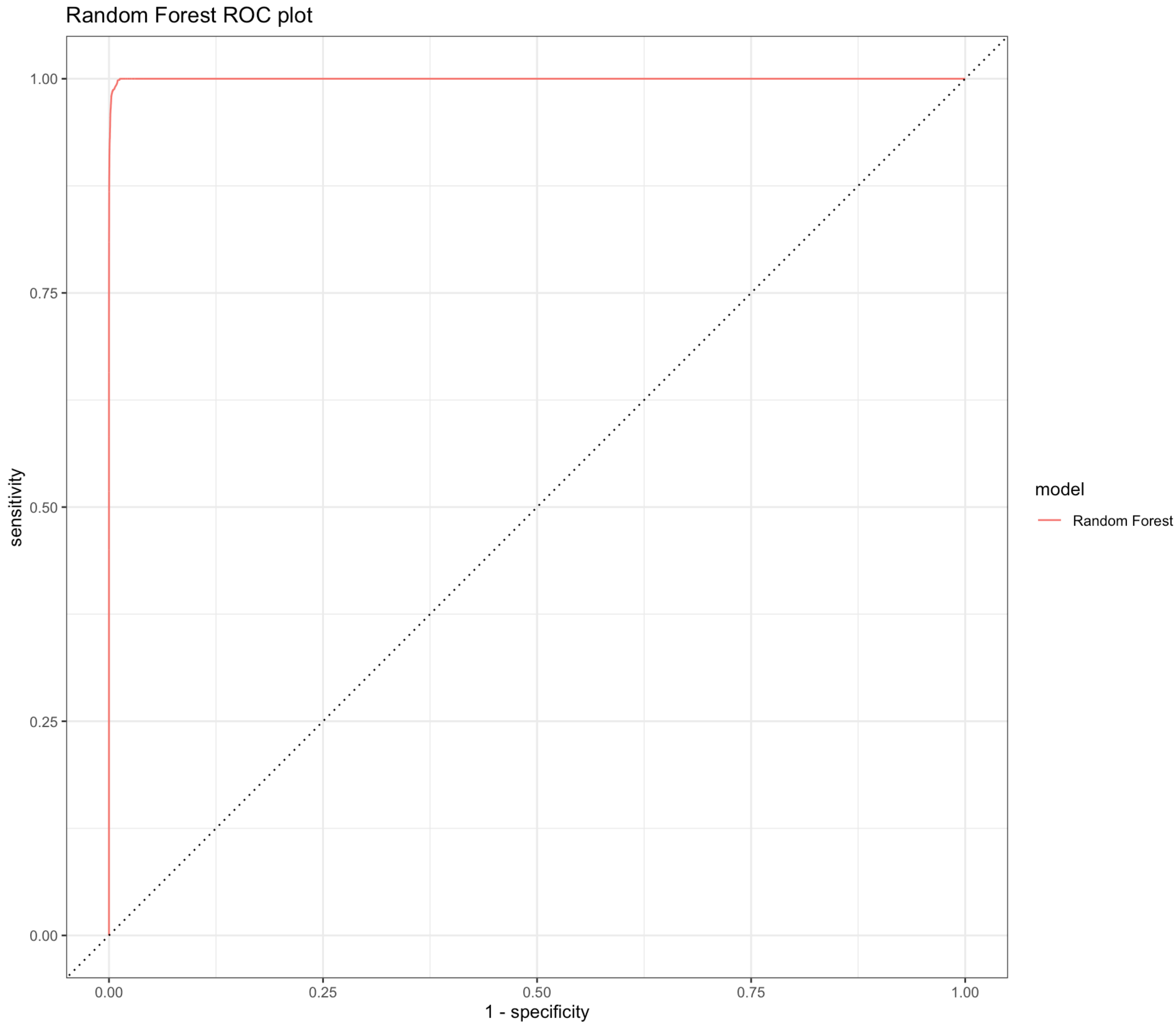
```
resamples <- vfold_cv(tuningtrain, v=10,strata=Class)
custom_metrics <- metric_set(roc_auc, accuracy)
cv_control <- control_resamples(save_pred=TRUE)

forest_cv <- fit_resamples(best_workflow, resamples, metrics=custom_metrics, control=cv_control)
```

```
cv_metrics <- bind_rows(
    collect_metrics(forest_cv) %>%
        mutate(model="Random Forest"),
)
cv_metrics %>%
    dplyr::select(model, .metric, mean) %>%
    pivot_wider(names_from=.metric, values_from=mean) %>%
    knitr::kable(caption="Random Forest Metrics", digits=5)
```

Random Forest Metrics

model	accuracy	roc_auc
Random Forest	0.99703	0.99974

```
bind_rows(
    collect_predictions(forest_cv) %>% mutate(model="Random Forest"),
) %>%
    group_by(model) %>%
    roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first") %>%
    autoplot()+labs(title="Random Forest ROC plot")
```




```
holdout<-holdout%>%
  mutate(
    Class=as.character(Class))
result_cvh <-  fit_resamples(best_workflow, resamples,
                            metrics=custom_metrics, control=cv_control)
fitted_model <- best_workflow %>% fit(holdout)
```

```
cv_metrics <- bind_rows(
    collect_metrics(result_cvh) %>%
        mutate(model="Random Forests"),
)
cv_metrics %>%
    dplyr::select(model, .metric, mean) %>%
    pivot_wider(names_from=.metric, values_from=mean) %>%
    knitr::kable(caption="Random Forests Metrics", digits=5)
```

Random Forests Metrics

model	accuracy	roc_auc
Random Forests	0.99693	0.99974

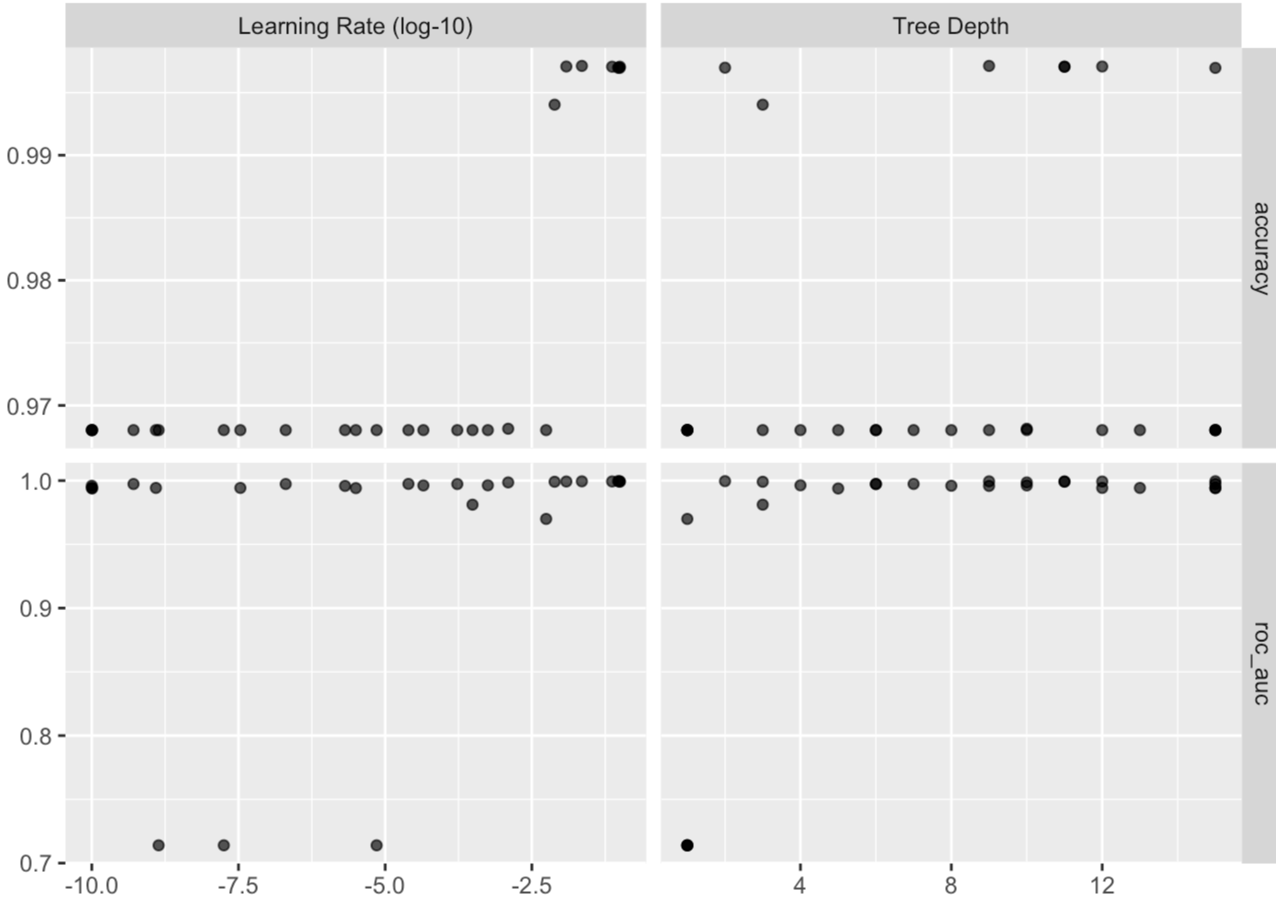
Xg Boost

```
library(bonsai) # this is required
boost_wf <- workflow() %>%
add_recipe(recipe(formula, data=Haiti_train)) %>%
add_model(boost_tree(mode="classification", engine="lightgbm",
trees=500, tree_depth=tune(), learn_rate=tune()))
```

```
parameters <- extract_parameter_set_dials(boost_wf)
tune_xgboost <- tune_bayes(boost_wf,
resamples=resamples,
metrics=custom_metrics,
param_info=parameters, iter=25)
```

```
## ! No improvement for 10 iterations; returning current results.
```

```
autoplot(tune_xgboost)
```




```
select_best(tune_xgboost,metric='roc_auc')
```

```
## # A tibble: 1 × 3
##   tree_depth learn_rate .config
##        <int>      <dbl> <chr>
## 1          2     0.0932 Iter11
```

```
best_boost_wf <- boost_wf %>%
  finalize_workflow(select_best(tune_xgboost, metric='roc_auc'))
```

```
boost_cv <- fit_resamples(best_boost_wf, resamples, metrics=custom_metrics, control=cv_c
ontrol)
```

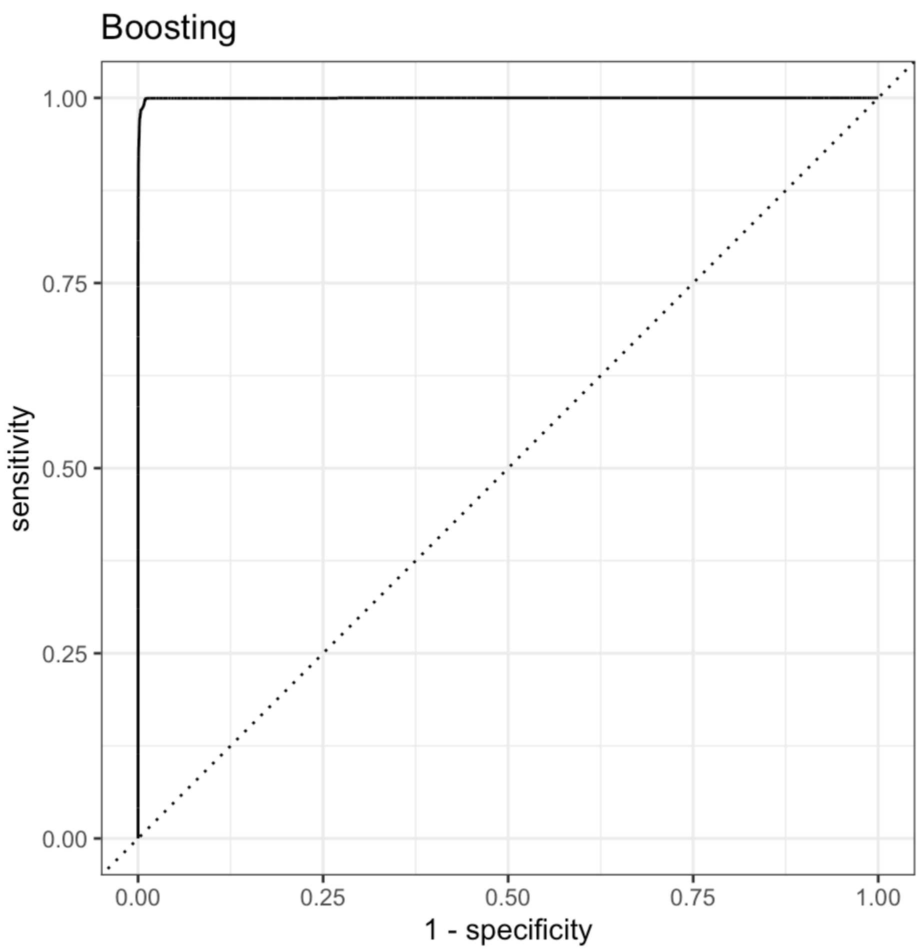
```
cv_metrics <- bind_rows(
    collect_metrics(boost_cv) %>%
        mutate(model="Xg Boost Holdout"),
)
cv_metrics %>%
    dplyr::select(model, .metric, mean) %>%
    pivot_wider(names_from=.metric, values_from=mean) %>%
    knitr::kable(caption="Xg Boost  Holdout Metrics", digits=5)
```

Xg Boost Holdout Metrics

model	accuracy	roc_auc
Xg Boost Holdout	0.997	0.9996

```
roc_cv_plot <- function(model_cv, model_name) {
  cv_predictions <- collect_predictions(model_cv)
  cv_roc <- cv_predictions %>%
    roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first")
return(autoplot(cv_roc) + labs(title=model_name))
}
```

```
roc_boost <- roc_cv_plot(boost_cv, "Boosting")
roc_boost
```



Penalized Logistic Regression (elastic net penalty):

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

recipe_spec <- recipe(formula, data=tuningtrain) %>%
    step_dummy(all_nominal(), -all_outcomes())
```

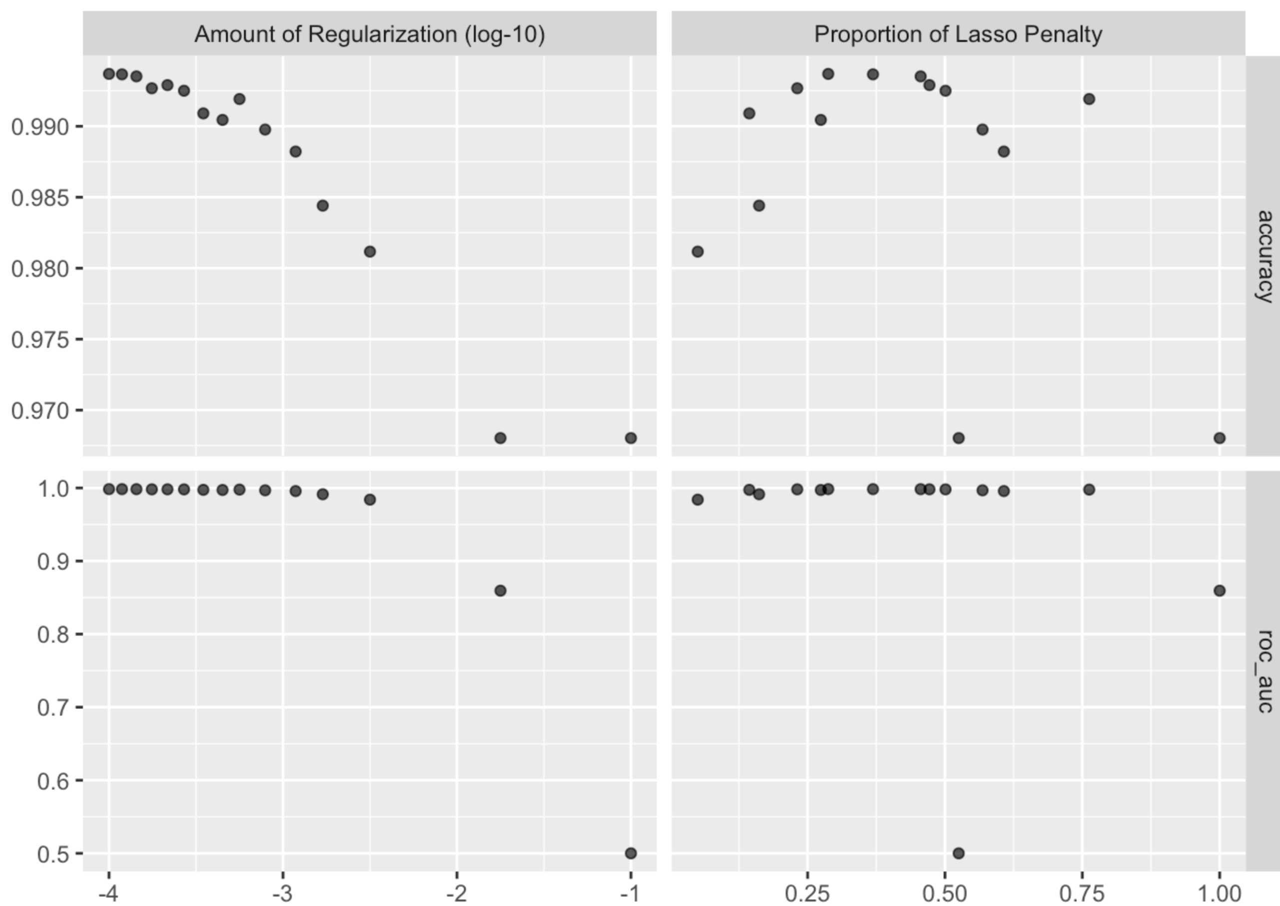
```
model_glmnet <- logistic_reg(engine="glmnet", mode="classification",
                            penalty=tune(), mixture=tune())
wf_penalized <- workflow() %>%
    add_model(model_glmnet) %>%
    add_recipe(recipe_spec)
```

```
parameters <- extract_parameter_set_dials(wf_penalized) %>%
    update(penalty=penalty(c(-4, -1)))
tune_results_penalized <- tune_grid(wf_penalized,
                           resamples=vfold_cv(Haiti_train),
                           grid=grid_regular(parameters))
```

```
tune_wf_penalized <- tune_bayes(wf_penalized, resamples=resamples, metrics=custom_metric
s,
                      param_info=parameters, iter=25)
```

```
## ! No improvement for 10 iterations; returning current results.
```

```
autoplot(tune_wf_penalized)
```




```
select_best(tune_wf_penalized, metric="roc_auc")
```

```
## # A tibble: 1 × 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1  0.0001   0.288 Preprocessor1_Model2
```

```
best_parameter_penalized <- select_best(tune_wf_penalized, metric="roc_auc")
best_wf_penalized <- finalize_workflow(wf_penalized, best_parameter_penalized)
```

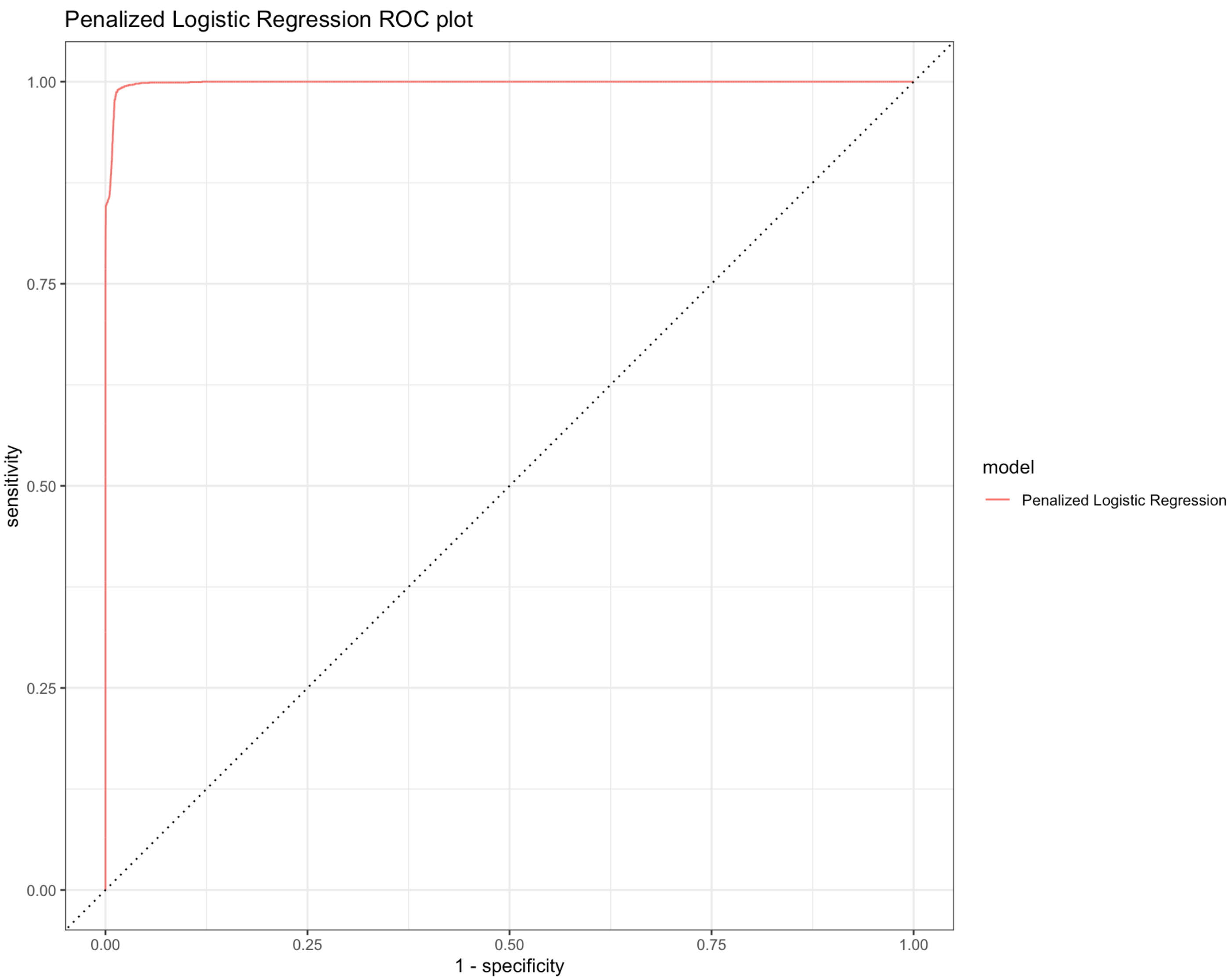
```
result_cv_penalized <-  fit_resamples(best_wf_penalized, resamples,
                            metrics=custom_metrics, control=cv_control)
fitted_model_penalized <- best_wf_penalized %>% fit(tuningtrain)
```

```
cv_metrics <- bind_rows(
    collect_metrics(result_cv_penalized) %>%
        mutate(model="Penalized Logistic Regression"),
)
cv_metrics %>%
    dplyr::select(model, .metric, mean) %>%
    pivot_wider(names_from=.metric, values_from=mean) %>%
    knitr::kable(caption="Penalized Logistic Regression Metrics", digits=5)
```

Penalized Logistic Regression Metrics

model	accuracy	roc_auc
Penalized Logistic Regression	0.99369	0.99841

```
bind_rows(
    collect_predictions(result_cv_penalized) %>% mutate(model="Penalized Logistic Regres
sion"),
) %>%
    group_by(model) %>%
    roc_curve(truth=Class, .pred_Blue_Tarp, event_level="first") %>%
    autoplot()+labs(title="Penalized Logistic Regression ROC plot")
```


Penalized Logistic Regression ROC plot
1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 - specificity
model
Penalized Logistic Regression