

Model

Eduardo Yuki Yada

Imports

```
library(tidyverse)
library(yaml)
library(tidymodels)
library(usemodels)
library(vip)
```

Loading data

```
load('../dataset/processed_data.RData')
load('../dataset/processed_dictionary.RData')

columns_list <- yaml.load_file("../auxiliar/columns_list.yaml")

outcome_column <- params$outcome_column
```

Filtering eligible patients

```
df = df %>%
  filter(disch_outcomes_t0 == 0)

df %>% dim
```

```
## [1] 15766 236
```

Eligible features

```
eligible_columns = df_names %>%
  filter(momento.aquisicao == 'Admissão t0') %>%
  .$variable.name

exception_columns = c('disch_outcomes_t0', 'icu_post_t0')

correlated_columns = c('year_procedure_1', # com year_adm_t0
  'age_surgery_1', # com age
  'admission_pre_t0_count', # com admission_t0
  'atb', # com meds_antimicrobianos
  'classe_meds_cardio_qtde', # com classe_meds_qtde
  'suporte_hemod' # com proced_invasivos_qtde
)

features = eligible_columns %>%
  base::intersect(c(columns_list$categorical_columns, columns_list$numerical_columns)) %>%
  setdiff(c(exception_columns, correlated_columns))

length(features)
```

```
## [1] 119
```

Train test split (70%/30%)

```
set.seed(42)

df[columns_list$outcome_columns] <- lapply(df[columns_list$outcome_columns], factor)

df_split <- initial_split(df %>% dplyr::select(all_of(c(features, outcome_column))),
                          prop = .7, strata = all_of(outcome_column))
df_train <- training(df_split)
df_test <- testing(df_split)

dim(df_train)[1] / dim(df)[1]

## [1] 0.6999873
dim(df_test)[1] / dim(df)[1]

## [1] 0.3000127
```

Imputation

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

library(doParallel)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)

imputation_recipe <- recipe(sprintf("%s ~ .", outcome_column) %>% as.formula,
                             data = df_train) %>%
  step_zv(all_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_impute_mode(all_nominal_predictors()) %>%
  step_impute_mean(all_numeric_predictors())
  # step_impute_knn(all_predictors(), neighbors = 3)

trained_imputation_recipe <- prep(imputation_recipe,
                                  training = df_train)

df_train_imputed <- bake(trained_imputation_recipe, df_train)
df_test_imputed <- bake(trained_imputation_recipe, df_test)
```

Functions

```
validation = function(model_fit, new_data) {
  library(pROC)
  library(caret)

  test_predictions_prob <-
    predict(model_fit, new_data, type = "prob") %>%
    rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
    .$`1`

  pROC_obj <- roc(
    new_data[[outcome_column]],
    test_predictions_prob,
    smoothed = TRUE,
    # arguments for ci
  )
}
```

```

    ci = TRUE,
    ci.alpha = 0.9,
    stratified = FALSE,
    # arguments for plot
    plot = TRUE,
    auc.polygon = TRUE,
    max.auc.polygon = TRUE,
    grid = TRUE,
    print.auc = TRUE,
    show.thres = TRUE
  )

sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type = "shape", col = "lightblue")
plot(sens.ci, type = "bars")

test_predictions_class <-
  predict(model_fit, new_data = new_data, type = "class") %>%
  rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
  .$class

conf_matrix = table(test_predictions_class, new_data[[outcome_column]])

confusionMatrix(conf_matrix) %>% print

return(pROC_obj)
}

```

XGBoost

```

xgboost_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_string2factor(one_of("sex", "race", "education_level", "patient_state",
    "underlying_heart_disease", "heart_disease", "nyha_basal", "hypertension",
    "prior_mi", "heart_failure", "af", "cardiac_arrest", "transplant", "valvopathy",
    "endocardites", "diabetes", "renal_failure", "hemodialysis", "stroke",
    "copd", "cancer", "surgery_count", "procedure_type_1", "reop_type_1", "cied_final_1",
    "death_intraop_1", "dialysis_hosp", "icu_hosp", "admission_pre_t0_180d", "ventilacao_mecanica")) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
  step_zv(all_predictors())

xgboost_spec <- boost_tree(
  trees = 500,
  tree_depth = tune(),
  min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  mtry = tune(),
  learn_rate = tune()
) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

xgboost_grid <- grid_latin_hypercube(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), df_train),

```

```

  learn_rate(),
  size = 30
)

set.seed(234)
df_folds <- vfold_cv(df_train, v = 5, strata = all_of(outcome_column))

xgboost_workflow <-
  workflow() %>%
  add_recipe(xgboost_recipe) %>%
  add_model(xgboost_spec)

xgboost_tune <-
  xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)

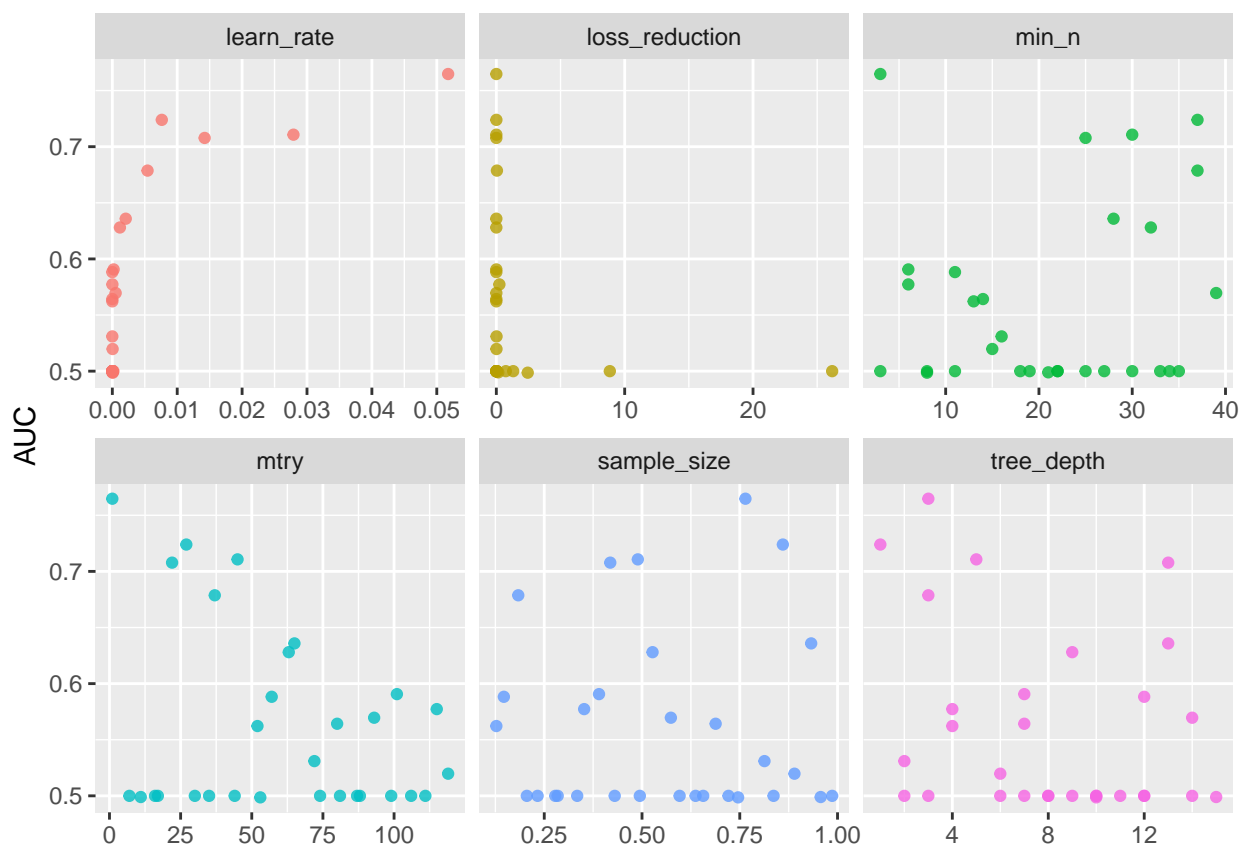
xgboost_tune %>%
  show_best("roc_auc")

## # A tibble: 5 x 12
##   mtry min_n tree_depth learn_rate loss_reduction sample_size
##   <int> <int>     <int>     <dbl>         <dbl>         <dbl>
## 1     1     3         3    0.0518         2.27e- 8         0.764
## 2    27    37         1    0.00763        3.34e- 3         0.860
## 3    45    30         5    0.0279        2.29e- 4         0.489
## 4    22    25        13    0.0142        1.28e-10         0.419
## 5    37    37         3    0.00543        4.72e- 2         0.184
## # ... with 6 more variables: .metric <chr>, .estimator <chr>,
## #   mean <dbl>, n <int>, std_err <dbl>, .config <chr>

best_xgboost <- xgboost_tune %>%
  select_best("roc_auc")

xgboost_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, mtry:sample_size) %>%
  pivot_longer(mtry:sample_size,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")

```



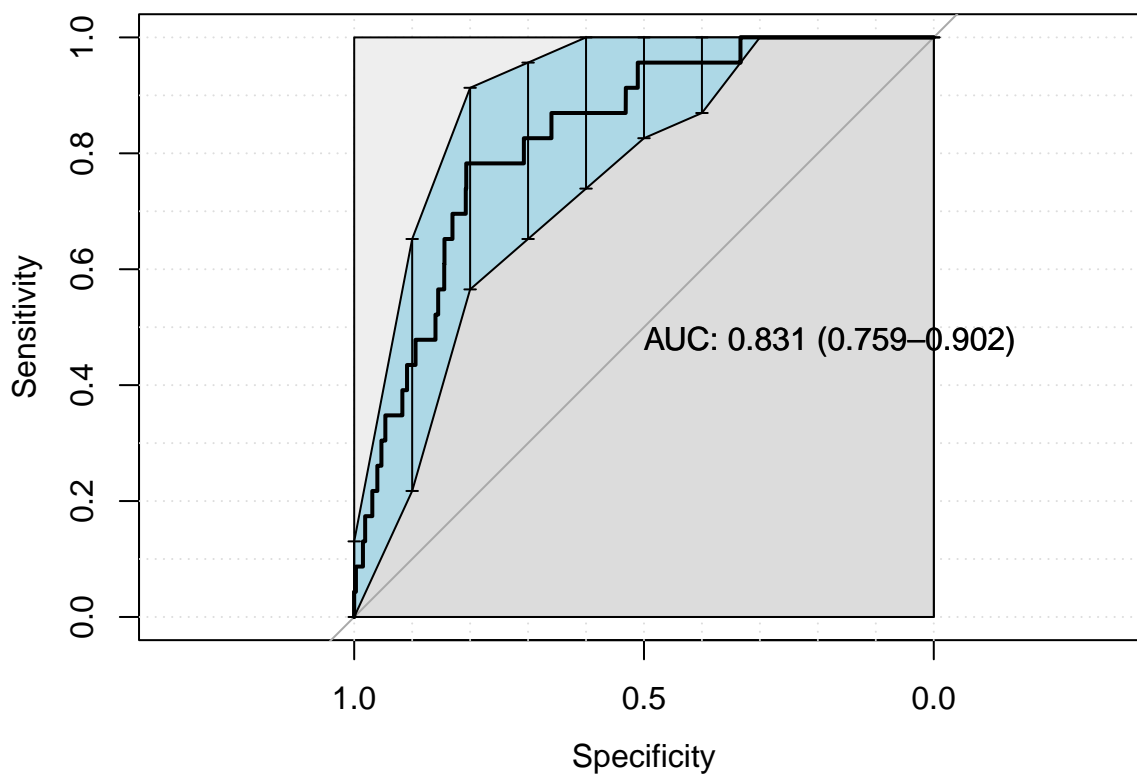
```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

last_xgboost_fit <-
  final_xgboost_workflow %>%
  last_fit(df_split)

final_xgboost_fit <- extract_workflow(last_xgboost_fit)

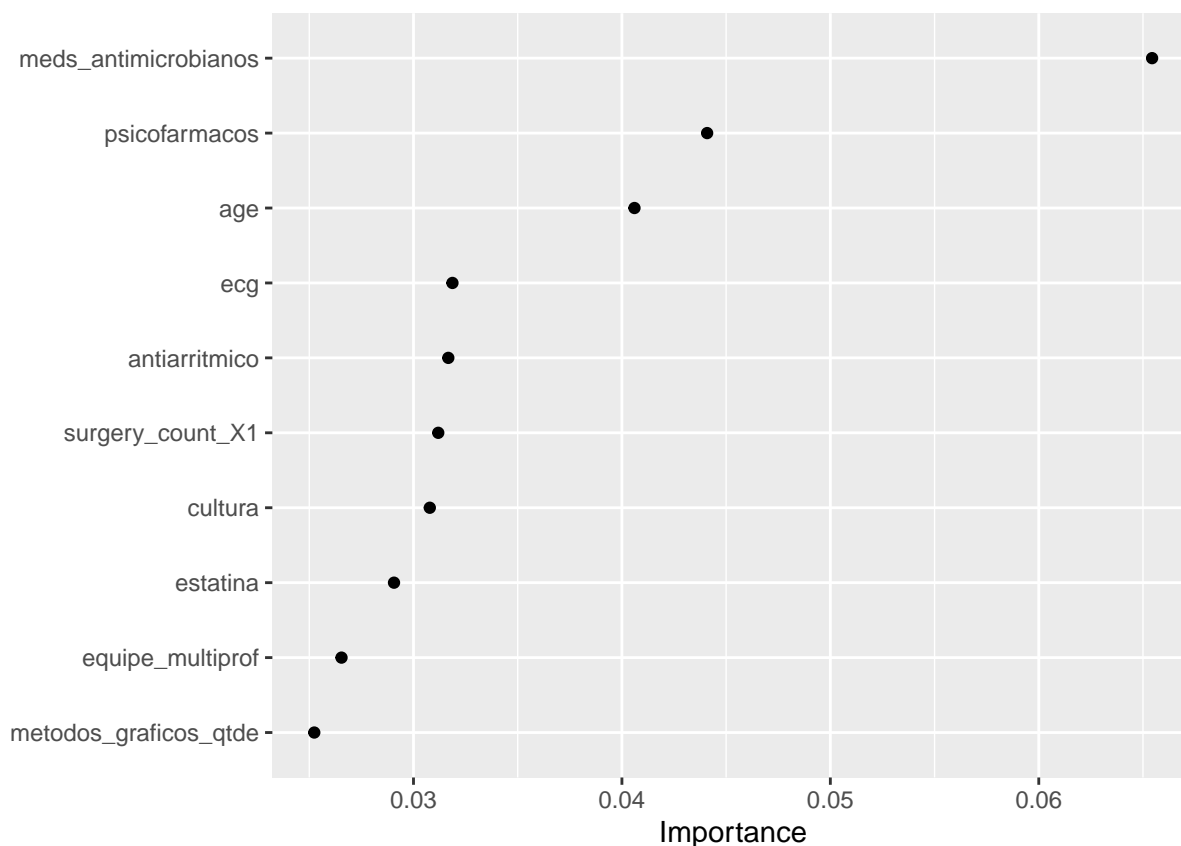
xgboost_auc = validation(final_xgboost_fit, df_test)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                0 4707   23
##                1    0    0
##
##              Accuracy : 0.9951
##              95% CI   : (0.9927, 0.9969)
##    No Information Rate : 0.9951
##    P-Value [Acc > NIR] : 0.5552
##
##              Kappa : 0
##
## Mcnemar's Test P-Value : 4.49e-06
##
##              Sensitivity : 1.0000
##              Specificity : 0.0000
##    Pos Pred Value : 0.9951
##    Neg Pred Value :   NaN
##    Prevalence : 0.9951
##    Detection Rate : 0.9951
##    Detection Prevalence : 1.0000
##    Balanced Accuracy : 0.5000
##
##    'Positive' Class : 0
##
```

```
final_xgboost_fit %>%
  fit(data = df_train) %>%
  extract_fit_parsnip() %>%
  vip(geom = "point")
```



GLM

```
glmnet_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_string2factor(one_of("sex", "race", "education_level", "patient_state",
    "underlying_heart_disease", "heart_disease", "nyha_basal", "hypertension",
    "prior_mi", "heart_failure", "af", "cardiac_arrest", "transplant", "valvopathy",
    "endocardites", "diabetes", "renal_failure", "hemodialysis", "stroke",
    "copd", "cancer", "surgery_count", "procedure_type_1", "reop_type_1", "cied_final_1",
    "death_intraop_1", "dialysis_hosp", "icu_hosp", "admission_pre_t0_180d", "ventilacao_mecanica")) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_numeric_predictors())

glmnet_spec <-
  logistic_reg(penalty = 0) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

glmnet_workflow <-
  workflow() %>%
  add_recipe(glmnet_recipe) %>%
  add_model(glmnet_spec)

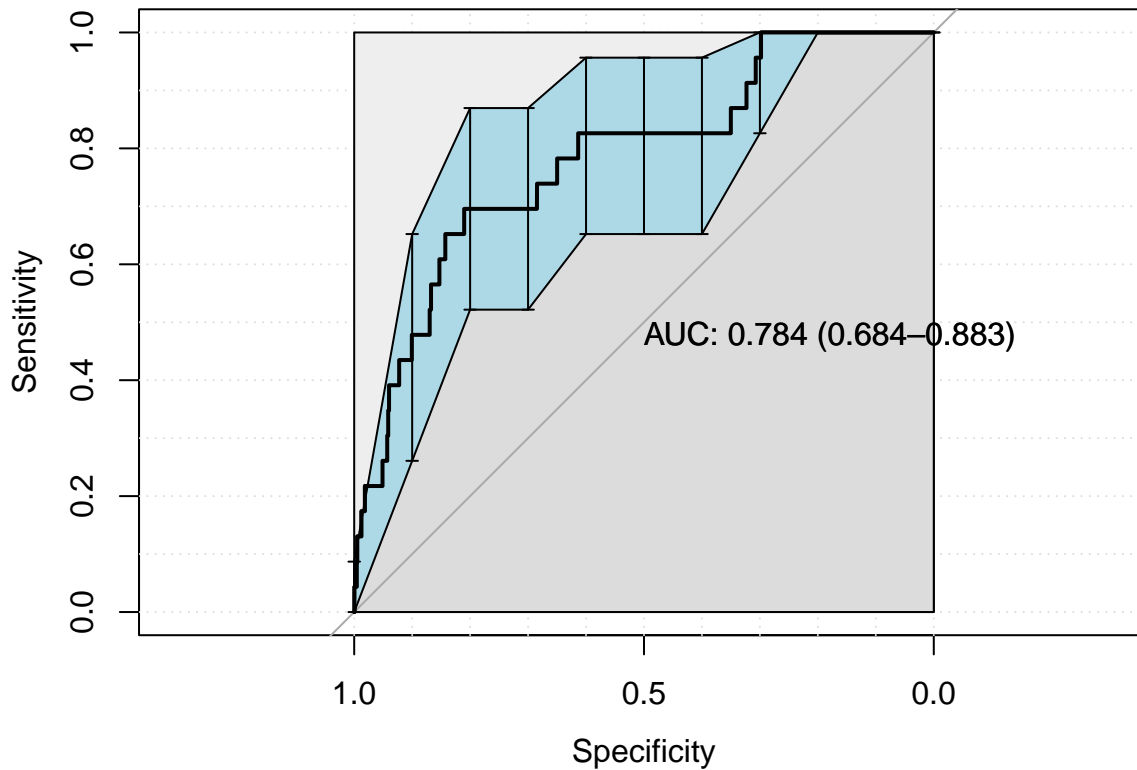
glm_fit <- glmnet_workflow %>%
  fit(df_train)

# glm_fit %>%
#   pull_workflow_fit() %>%
#   tidy()
```

```
glm_auc = validation(glm_fit, df_test)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
## Confusion Matrix and Statistics
```

```
##
```

```
##
```

```
## test_predictions_class    0    1
##          0 4707    23
##          1    0    0
```

```
##
```

```
##          Accuracy : 0.9951
```

```
##          95% CI : (0.9927, 0.9969)
```

```
##          No Information Rate : 0.9951
```

```
##          P-Value [Acc > NIR] : 0.5552
```

```
##
```

```
##          Kappa : 0
```

```
##
```

```
##          McNemar's Test P-Value : 4.49e-06
```

```
##
```

```
##          Sensitivity : 1.0000
```

```
##          Specificity : 0.0000
```

```
##          Pos Pred Value : 0.9951
```

```
##          Neg Pred Value : NaN
```

```
##          Prevalence : 0.9951
```

```
##          Detection Rate : 0.9951
```

```
##          Detection Prevalence : 1.0000
```

```
##          Balanced Accuracy : 0.5000
```

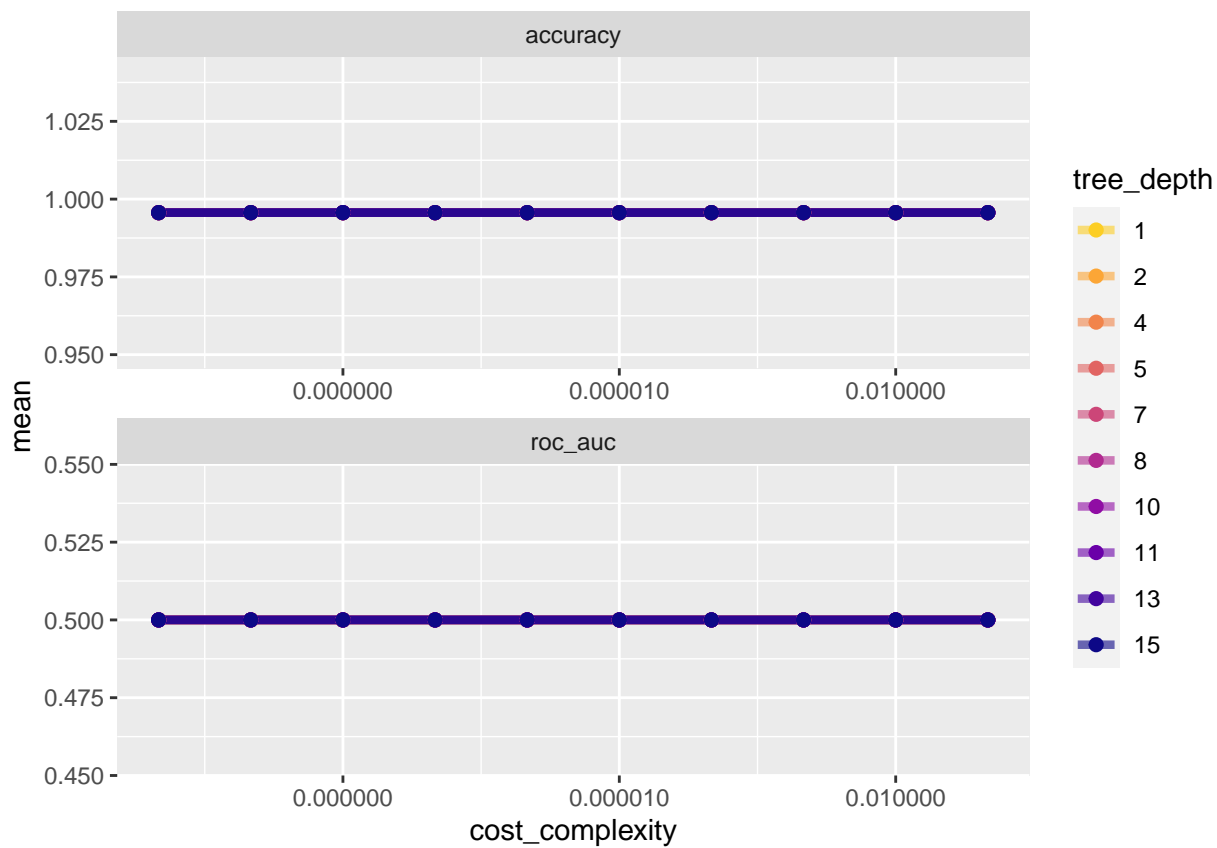
```
##
```

```
##          'Positive' Class : 0
```

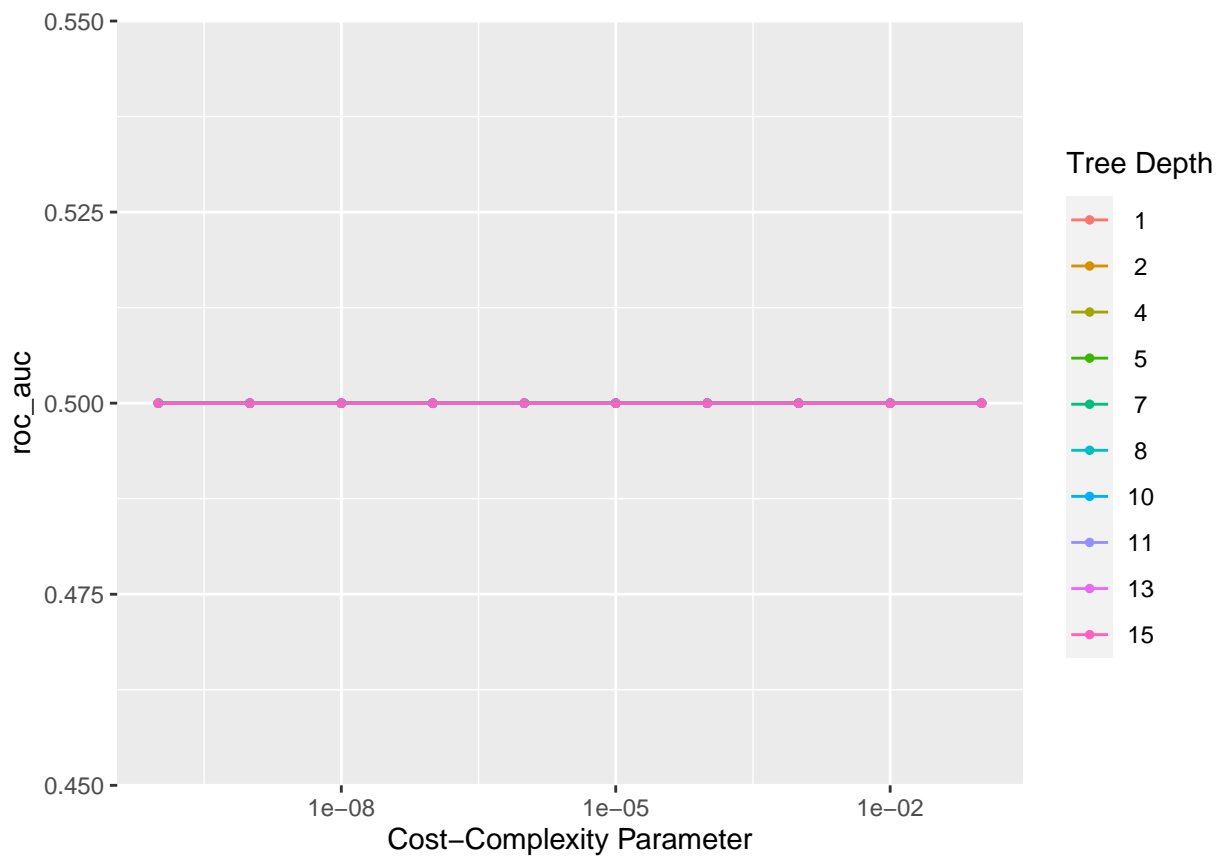
```
##
```


Decision Tree

```
tree_recipe <-  
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%  
  step_string2factor(one_of("sex", "race", "education_level", "patient_state",  
    "underlying_heart_disease", "heart_disease", "nyha_basal", "hypertension",  
    "prior_mi", "heart_failure", "af", "cardiac_arrest", "transplant", "valvopathy",  
    "endocardites", "diabetes", "renal_failure", "hemodialysis", "stroke",  
    "copd", "cancer", "surgery_count", "procedure_type_1", "reop_type_1", "cied_final_1",  
    "death_intraop_1", "dialysis_hosp", "icu_hosp", "admission_pre_t0_180d", "ventilacao_mecanica")) %>%  
  step_novel(all_nominal_predictors()) %>%  
  step_zv(all_predictors())  
  
tree_spec <-  
  decision_tree(cost_complexity = tune(),  
    tree_depth = tune()) %>%  
  set_mode("classification") %>%  
  set_engine("rpart")  
  
tree_grid <- grid_regular(cost_complexity(),  
  tree_depth(),  
  levels = 10)  
  
set.seed(234)  
df_folds <- vfold_cv(df_train, v = 5, strata = all_of(outcome_column))  
  
tree_workflow <-  
  workflow() %>%  
  add_recipe(tree_recipe) %>%  
  add_model(tree_spec)  
  
tree_tune <-  
  tree_workflow %>%  
  tune_grid(resamples = df_folds,  
    grid = tree_grid)  
  
tree_tune %>%  
  collect_metrics()  
  
tree_tune %>%  
  collect_metrics() %>%  
  mutate(tree_depth = factor(tree_depth)) %>%  
  ggplot(aes(cost_complexity, mean, color = tree_depth)) +  
  geom_line(size = 1.5, alpha = 0.6) +  
  geom_point(size = 2) +  
  facet_wrap(~ .metric, scales = "free", nrow = 2) +  
  scale_x_log10(labels = scales::label_number()) +  
  scale_color_viridis_d(option = "plasma", begin = .9, end = 0)
```



```
autoplot(tree_tune, metric = "roc_auc")
```



```
tree_tune %>%
  show_best("roc_auc")

best_tree <- tree_tune %>%
```

```

select_best("roc_auc")

final_tree_workflow <-
  tree_workflow %>%
  finalize_workflow(best_tree)

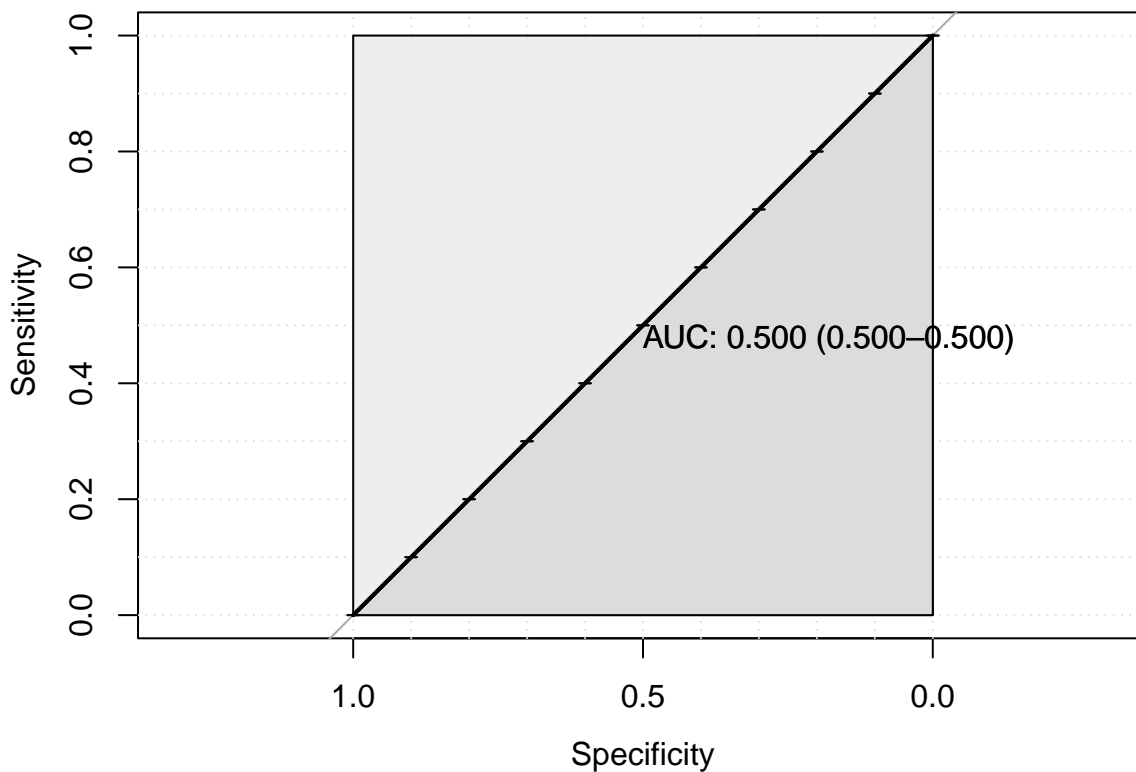
last_tree_fit <-
  final_tree_workflow %>%
  last_fit(df_split)

final_tree_fit <- extract_workflow(last_tree_fit)

tree_auc = validation(final_tree_fit, df_test)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

```



```

if (tree_auc$auc > 0.55){
  final_tree_fit %>%
    extract_fit_parsnip() %>%
    vip()
}

```

Random Forest

```

rf_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_string2factor(one_of("sex", "race", "education_level", "patient_state",
    "underlying_heart_disease", "heart_disease", "nyha_basal", "hypertension",
    "prior_mi", "heart_failure", "af", "cardiac_arrest", "transplant", "valvopathy",
    "endocarditis", "diabetes", "renal_failure", "hemodialysis", "stroke",
    "copd", "cancer", "surgery_count", "procedure_type_1", "reop_type_1", "cied_final_1",

```

```

    "death_intraop_1", "dialysis_hosp", "icu_hosp", "admission_pre_t0_180d", "ventilacao_mecanica")) %>%
step_novel(all_nominal_predictors()) %>%
step_zv(all_predictors()) %>%
step_dummy(all_nominal_predictors()) %>%
step_impute_mode(all_nominal_predictors()) %>%
step_impute_mean(all_numeric_predictors())

rf_spec <-
  rand_forest(mtry = tune(),
              trees = 1000,
              min_n = tune()) %>%
  set_mode("classification") %>%
  set_engine("ranger")

rf_grid <- grid_regular(mtry(range = c(1, 10)),
                       min_n(),
                       levels = 5)

set.seed(234)
df_folds <- vfold_cv(df_train, v = 5, strata = all_of(outcome_column))

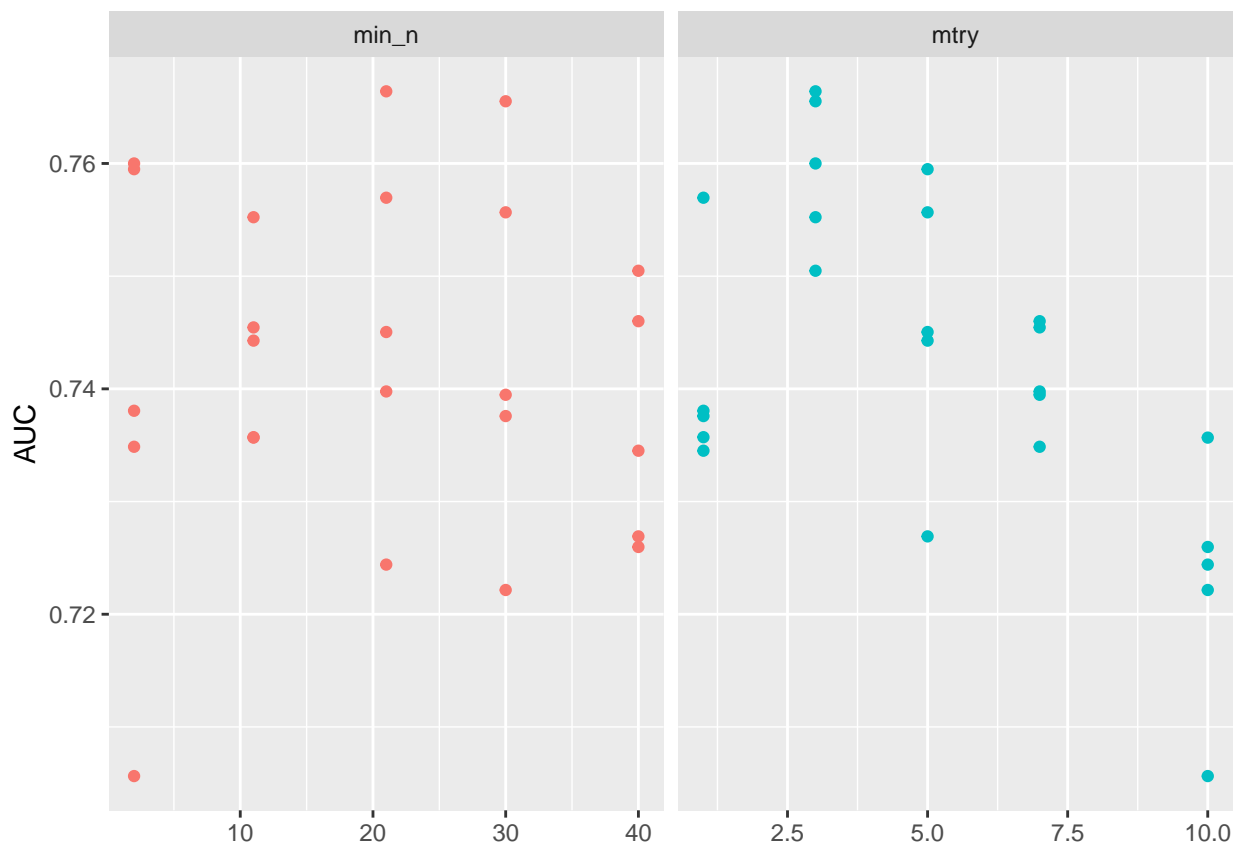
rf_workflow <-
  workflow() %>%
  add_recipe(rf_recipe) %>%
  add_model(rf_spec)

rf_tune <-
  rf_workflow %>%
  tune_grid(resamples = df_folds,
            grid = rf_grid)

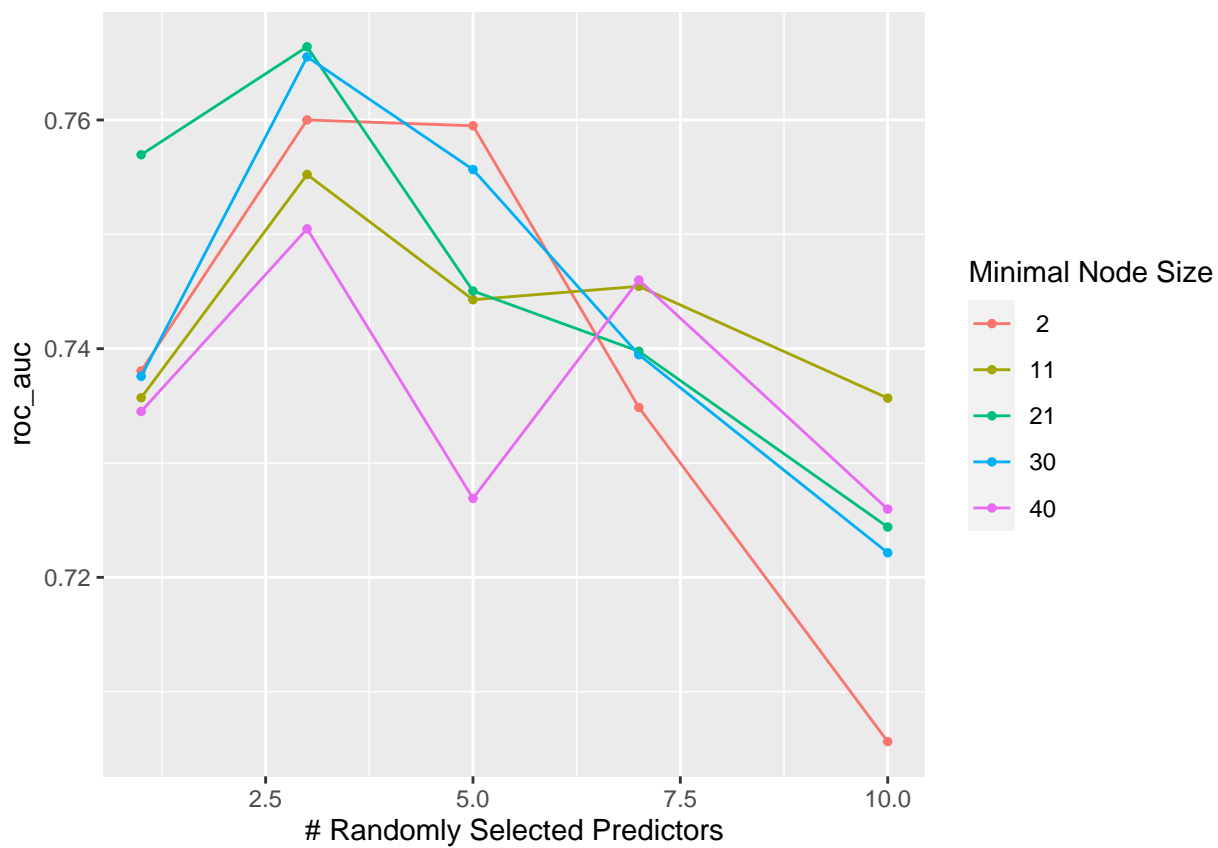
rf_tune %>%
  collect_metrics()

rf_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, min_n, mtry) %>%
  pivot_longer(min_n:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")

```



```
autoplot(rf_tune, metric = "roc_auc")
```



```
rf_tune %>%
  show_best("roc_auc")

best_rf <- rf_tune %>%
```

```

select_best("roc_auc")

final_rf_workflow <-
  rf_workflow %>%
  finalize_workflow(best_rf)

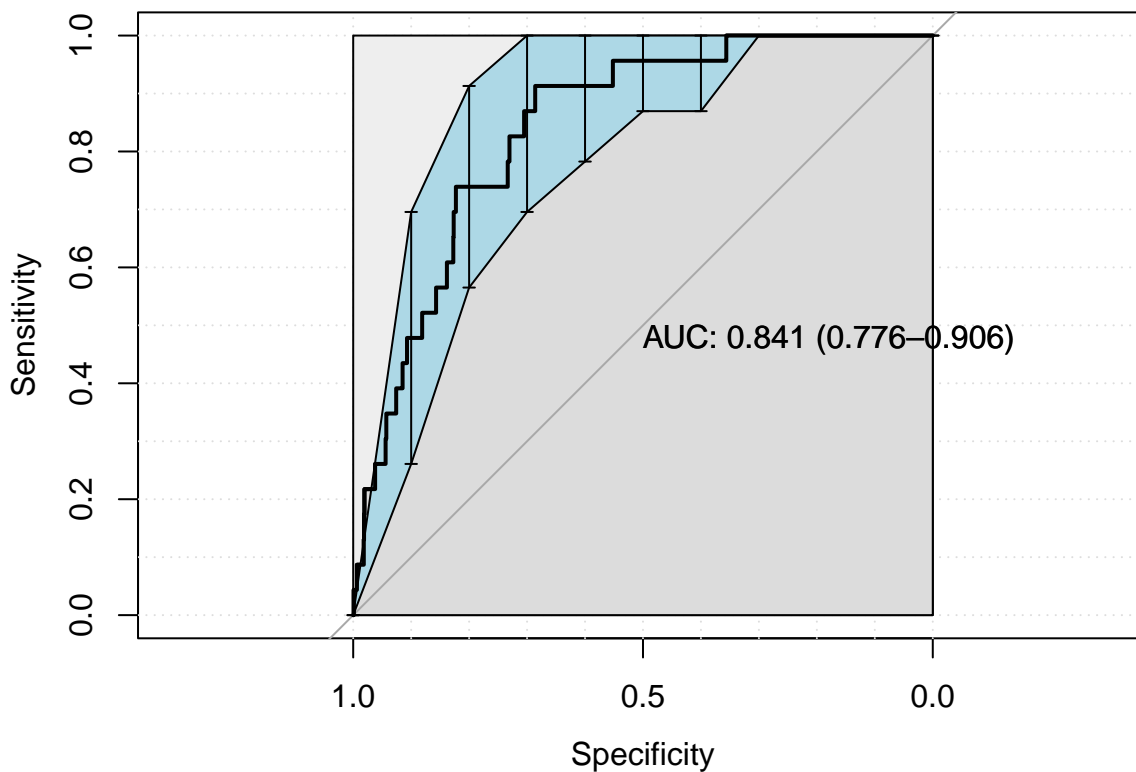
last_rf_fit <-
  final_rf_workflow %>%
  last_fit(df_split)

final_rf_fit <- extract_workflow(last_rf_fit)

rf_auc = validation(final_rf_fit, df_test)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

```



```

# final_rf_fit %>%
#   extract_fit_parsnip() %>%
#   vip()

```

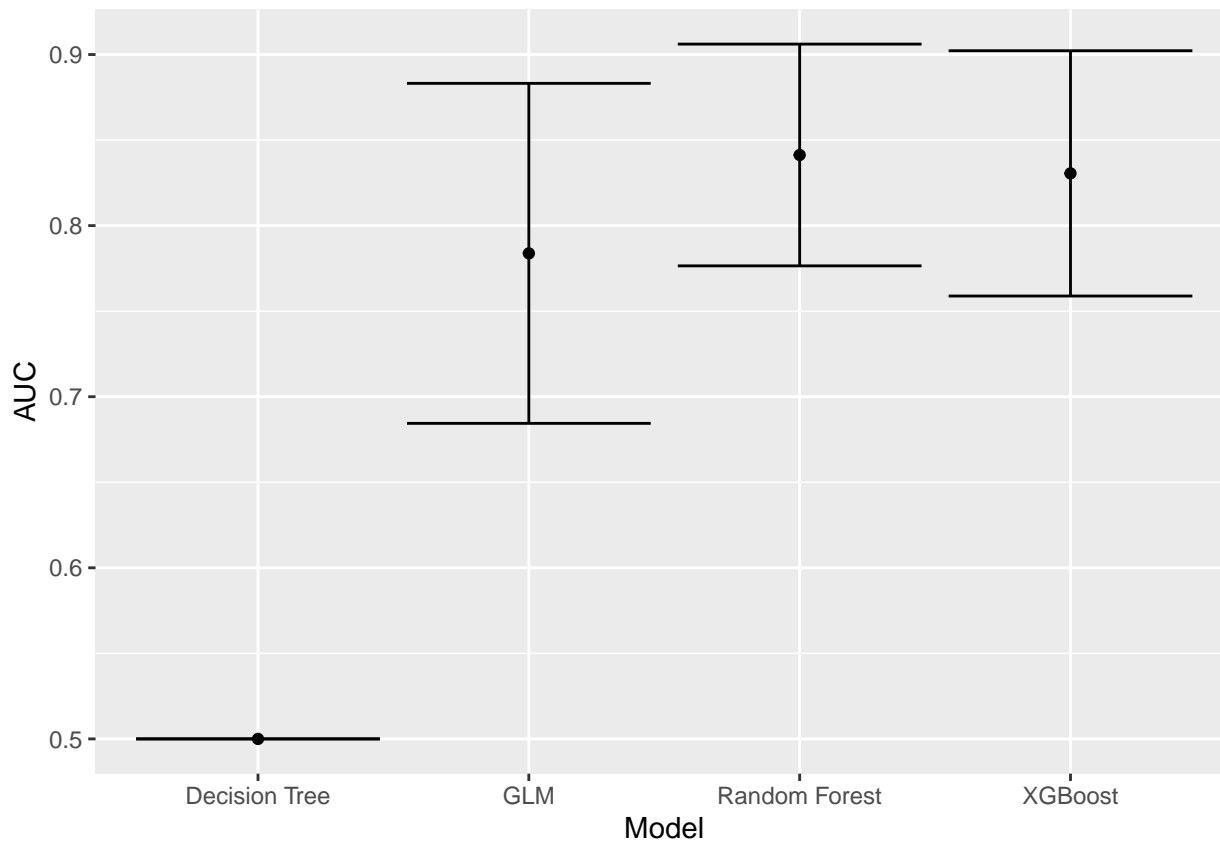
Models Comparison

```

df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'XGBoost', as.numeric(xgboost_auc$auc), xgboost_auc$ci[1], xgboost_auc$ci[3],
  'GLM', as.numeric(glm_auc$auc), glm_auc$ci[1], glm_auc$ci[3],
  'Decision Tree', as.numeric(tree_auc$auc), tree_auc$ci[1], tree_auc$ci[3],
  'Random Forest', as.numeric(rf_auc$auc), rf_auc$ci[1], rf_auc$ci[3]
) %>%
  mutate(Target = outcome_column)

```

```
df_auc %>%
  ggplot(aes(x = Model, y = AUC, ymin = `Lower Limit`, ymax = `Upper Limit`)) +
    geom_point() +
    geom_errorbar()
```



```
saveRDS(df_auc, sprintf("../EDA/auxiliar/performance/%s_auc_result.RData", outcome_column))
```