

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
```

Global parameters

```
k = 4 # Number of folds for cross validation

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

Functions

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

  test_predictions_prob <-
    predict(model_fit, new_data = 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)
}

```

Boosted Tree (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
)

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

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

```



```
## ! Fold4: preprocessor 1/1, model 30/30 (predictions): There are new levels in a factor: NA
```

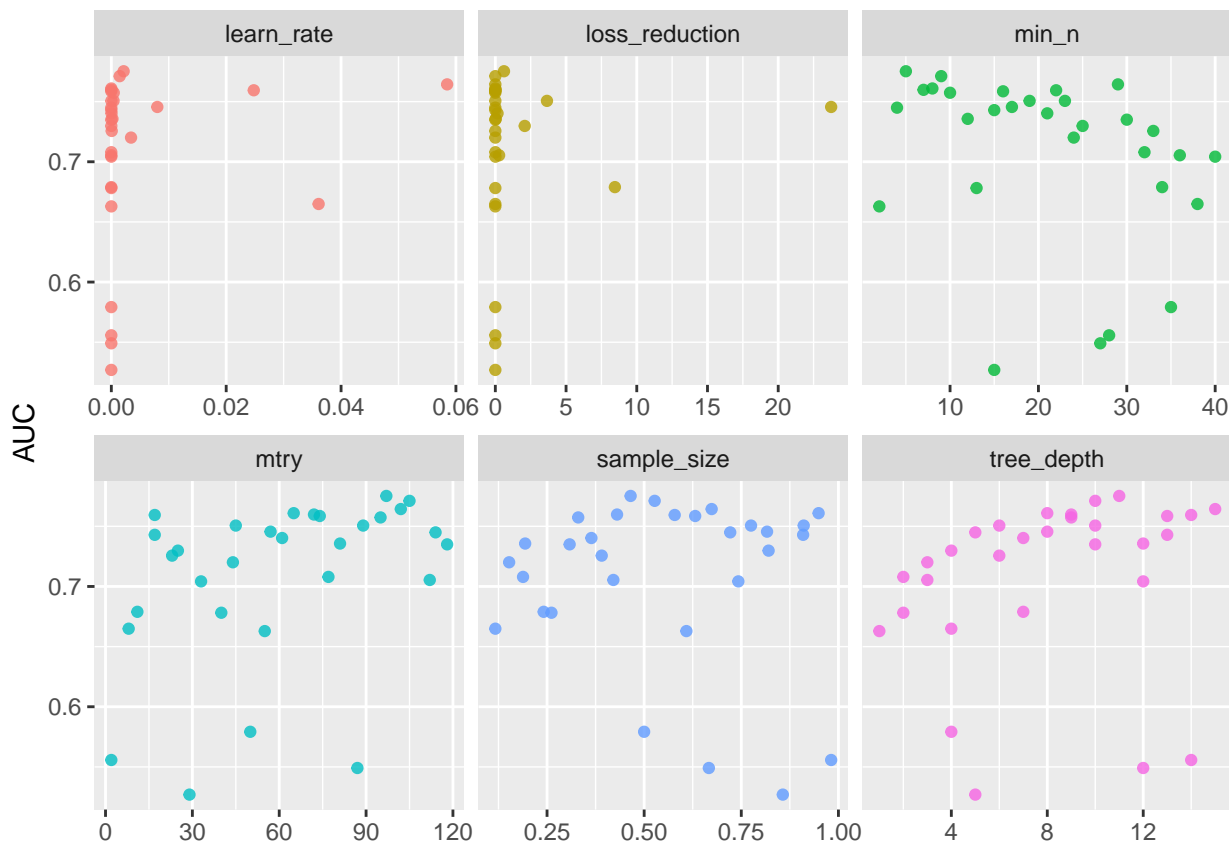
```
xgboost_tune %>%
  show_best("roc_auc")
```

```
## # A tibble: 5 x 12
```

```
##   mtry min_n tree_depth learn_rate loss_reduction sample_size .metric .estimator mean
##   <int> <int>    <int>    <dbl>      <dbl>      <dbl> <chr>   <chr>   <dbl>
## 1    97     5        11 0.00215      0.611        0.465 roc_auc binary  0.775
## 2   105     9         10 0.00146      0.0000000254 0.527 roc_auc binary  0.771
## 3   102    29         15 0.0585       0.00000000640 0.674 roc_auc binary  0.764
## 4    65     8         8 0.000000225 0.00119        0.949 roc_auc binary  0.761
## 5    72     7         9 0.00000204 0.0156        0.430 roc_auc binary  0.760
## # ... with 3 more variables: 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)
```

```
## ! train/test split: preprocessor 1/1: There are new levels in a factor: NA
```

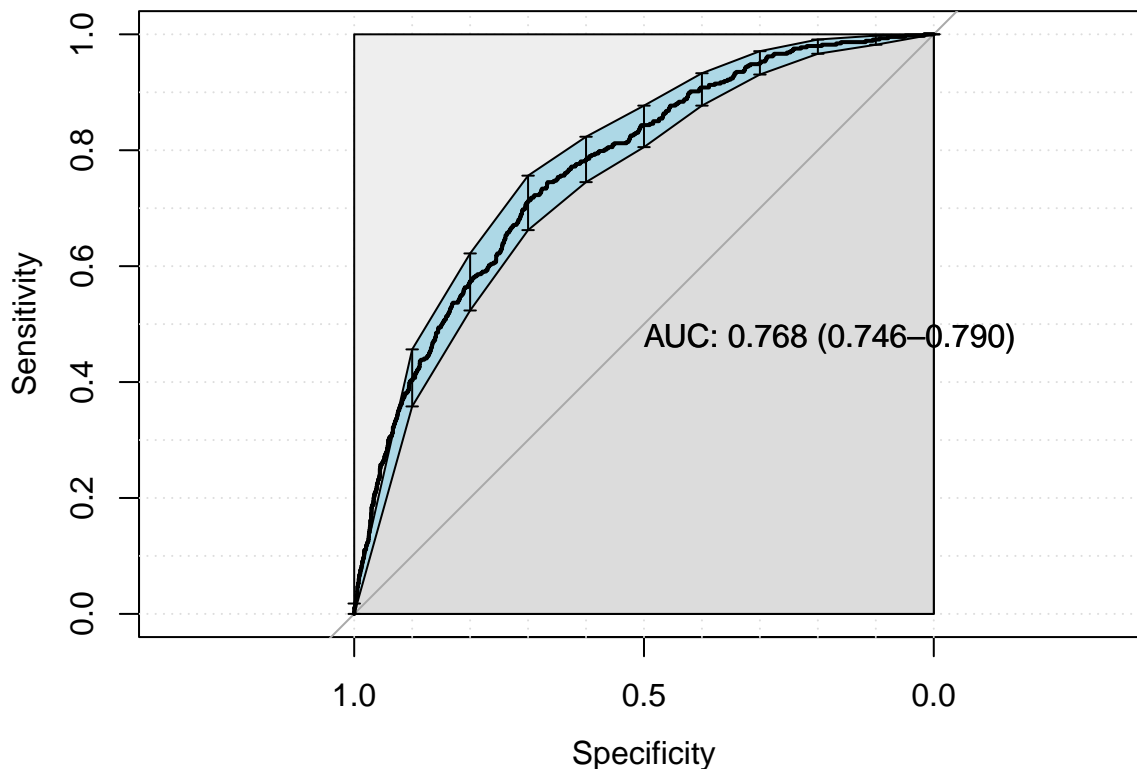
```
## ! train/test split: preprocessor 1/1, model 1/1 (predictions): There are new levels in a factor: NA
```

```
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 4279  442
```

```
##           1    4    5
```

```
##
```

```
##           Accuracy : 0.9057
```

```
##           95% CI : (0.897, 0.9139)
```

```
##           No Information Rate : 0.9055
```

```
##           P-Value [Acc > NIR] : 0.4928
```

```
##
```

```
##           Kappa : 0.0183
```

```
##
```

```
##           McNemar's Test P-Value : <2e-16
```

```
##
```

```
##           Sensitivity : 0.99907
```

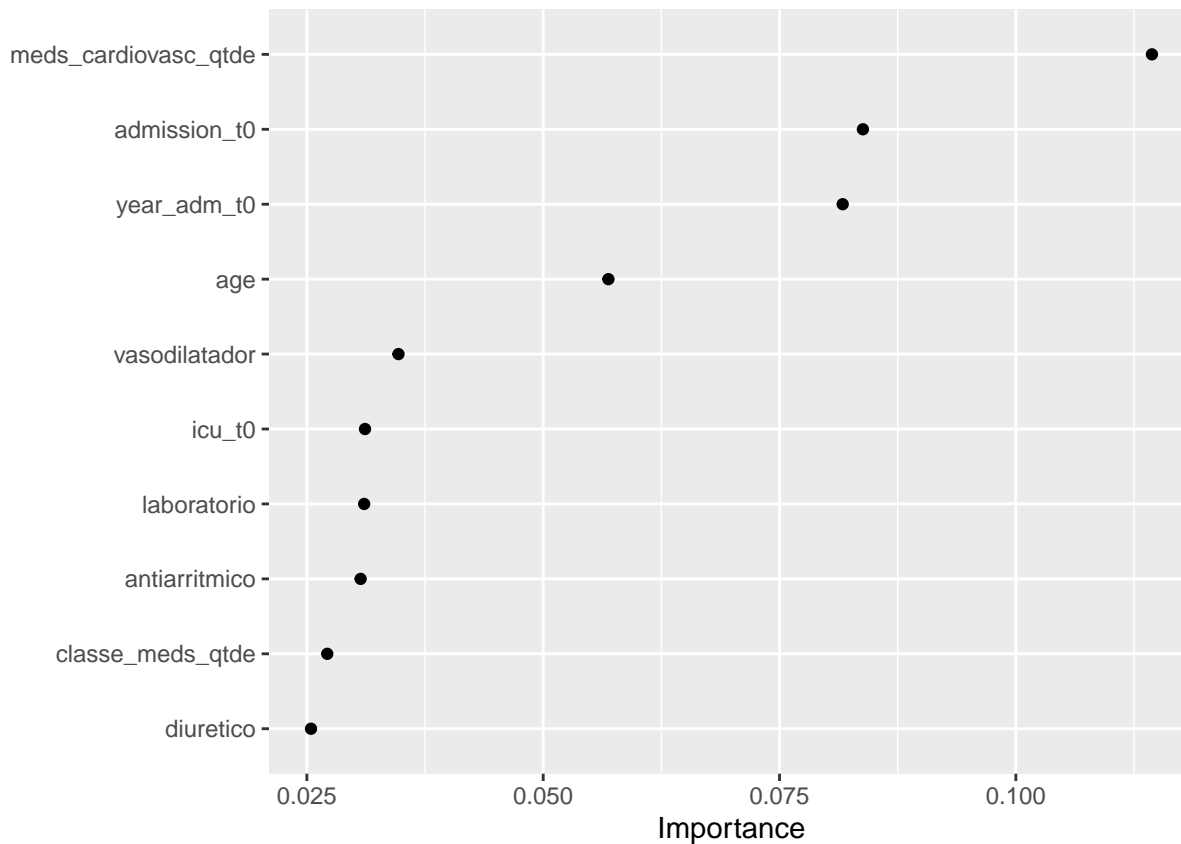
```
##           Specificity : 0.01119
```

```
##           Pos Pred Value : 0.90638
```



```
##      Neg Pred Value : 0.55556
##      Prevalence : 0.90550
##      Detection Rate : 0.90465
##      Detection Prevalence : 0.99810
##      Balanced Accuracy : 0.50513
##
##      '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()

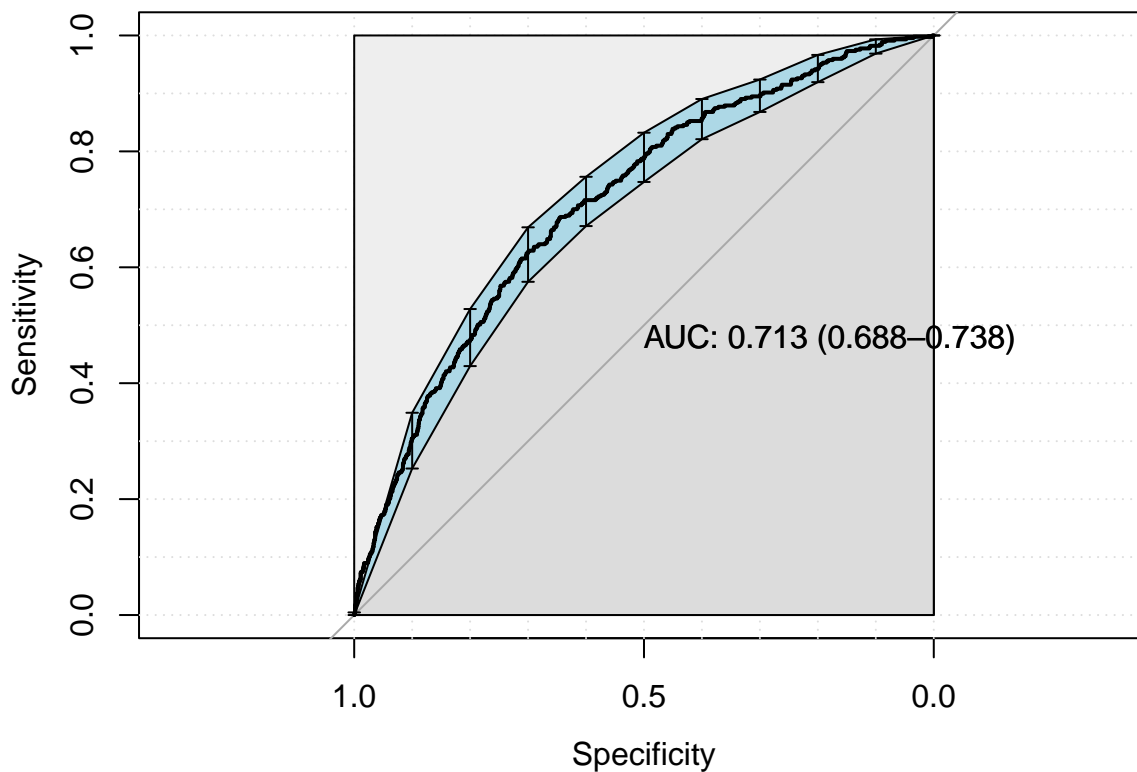
glm_auc = validation(glm_fit, df_test)

```

```

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

```



```

## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##          0 4269  436
##          1   13   11
##
##          Accuracy : 0.9051
##          95% CI   : (0.8963, 0.9133)
##    No Information Rate : 0.9055
##    P-Value [Acc > NIR] : 0.5521
##
##          Kappa   : 0.0374
##
##  McNemar's Test P-Value : <2e-16
##

```

```
##           Sensitivity : 0.99696
##           Specificity : 0.02461
##           Pos Pred Value : 0.90733
##           Neg Pred Value : 0.45833
##           Prevalence : 0.90548
##           Detection Rate : 0.90273
##           Detection Prevalence : 0.99492
##           Balanced Accuracy : 0.51079
##
##           '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")

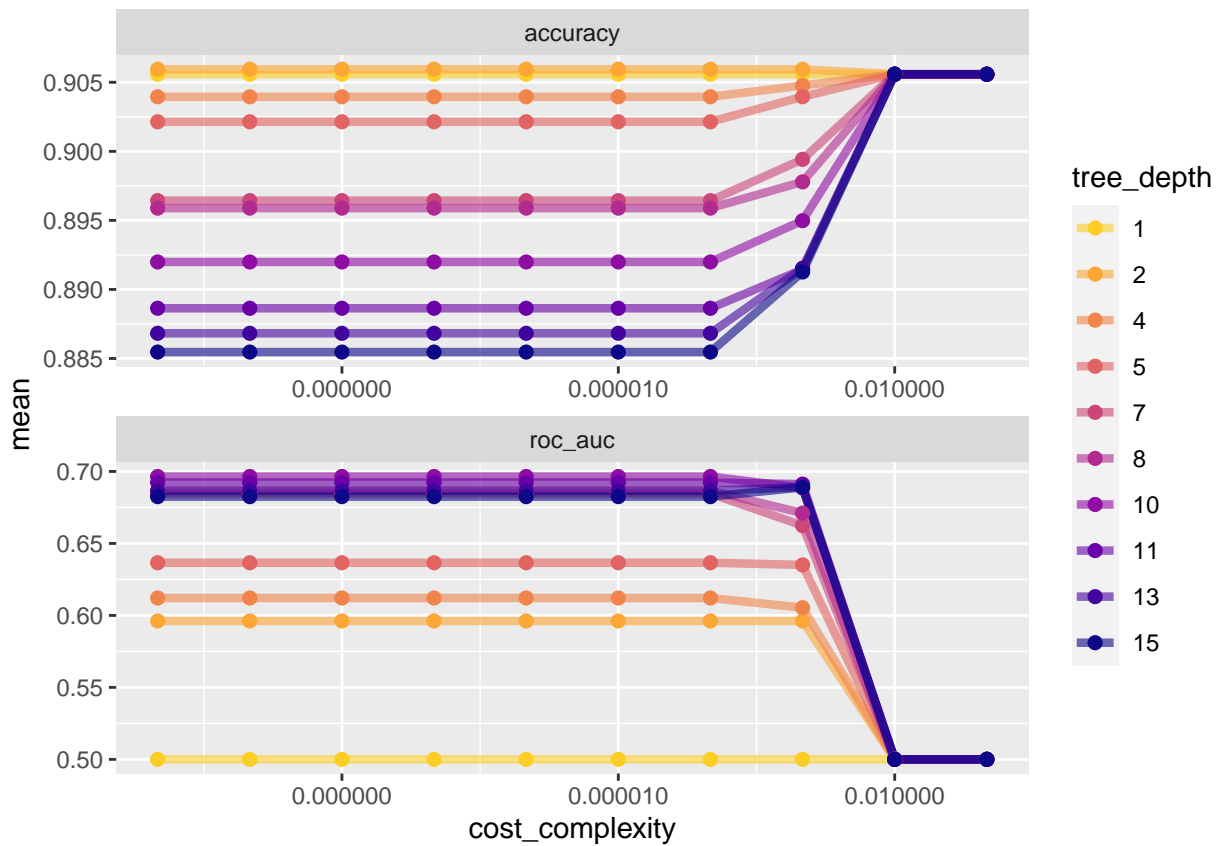
tree_grid <- grid_regular(cost_complexity(),
  tree_depth(),
  levels = 10)

tree_workflow <-
  workflow() %>%
  add_recipe(tree_recipe) %>%
  add_model(tree_spec)

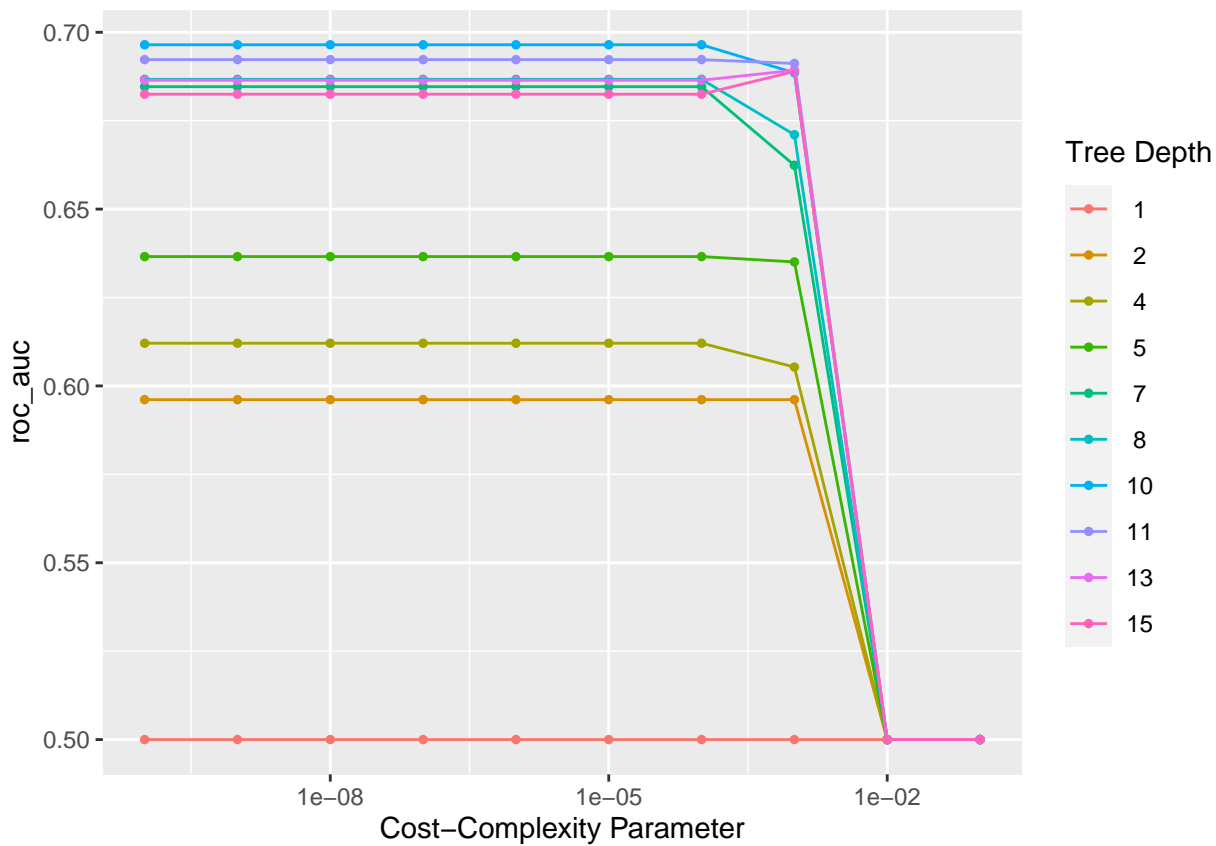
tree_tune <-
  tree_workflow %>%
  tune_grid(resamples = df_folds,
    grid = tree_grid)

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)

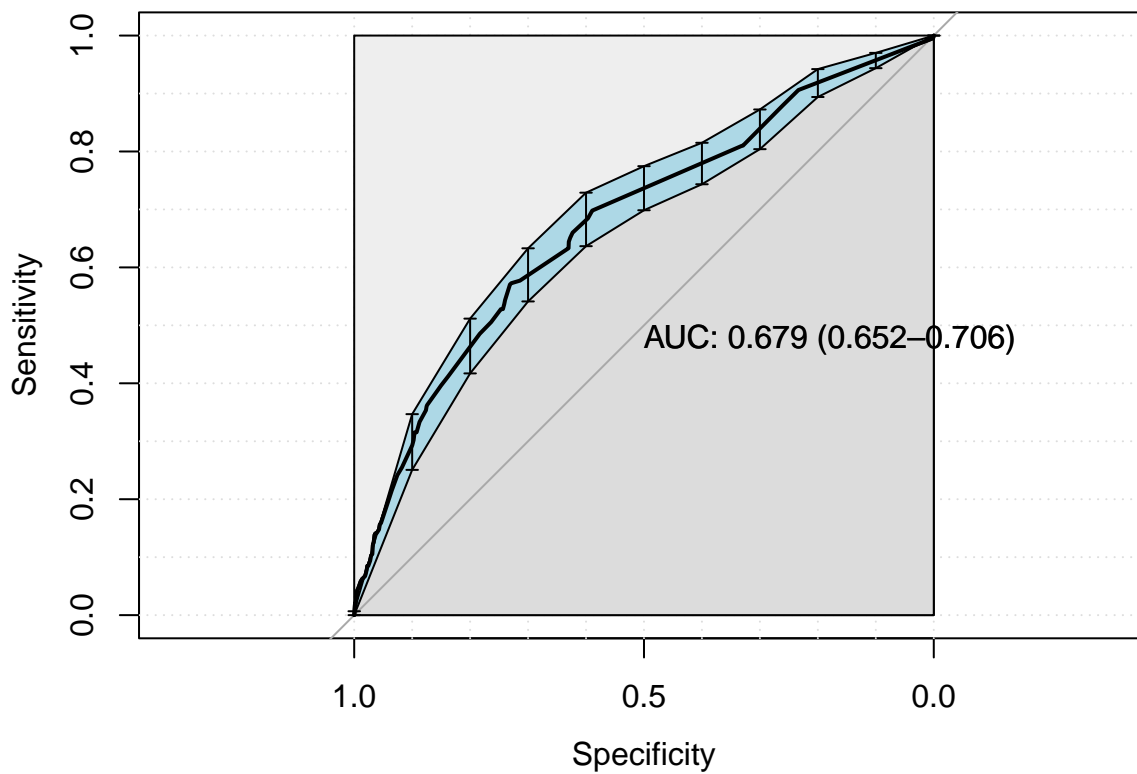
last_tree_fit <-
  final_tree_workflow %>%
  last_fit(df_split)

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

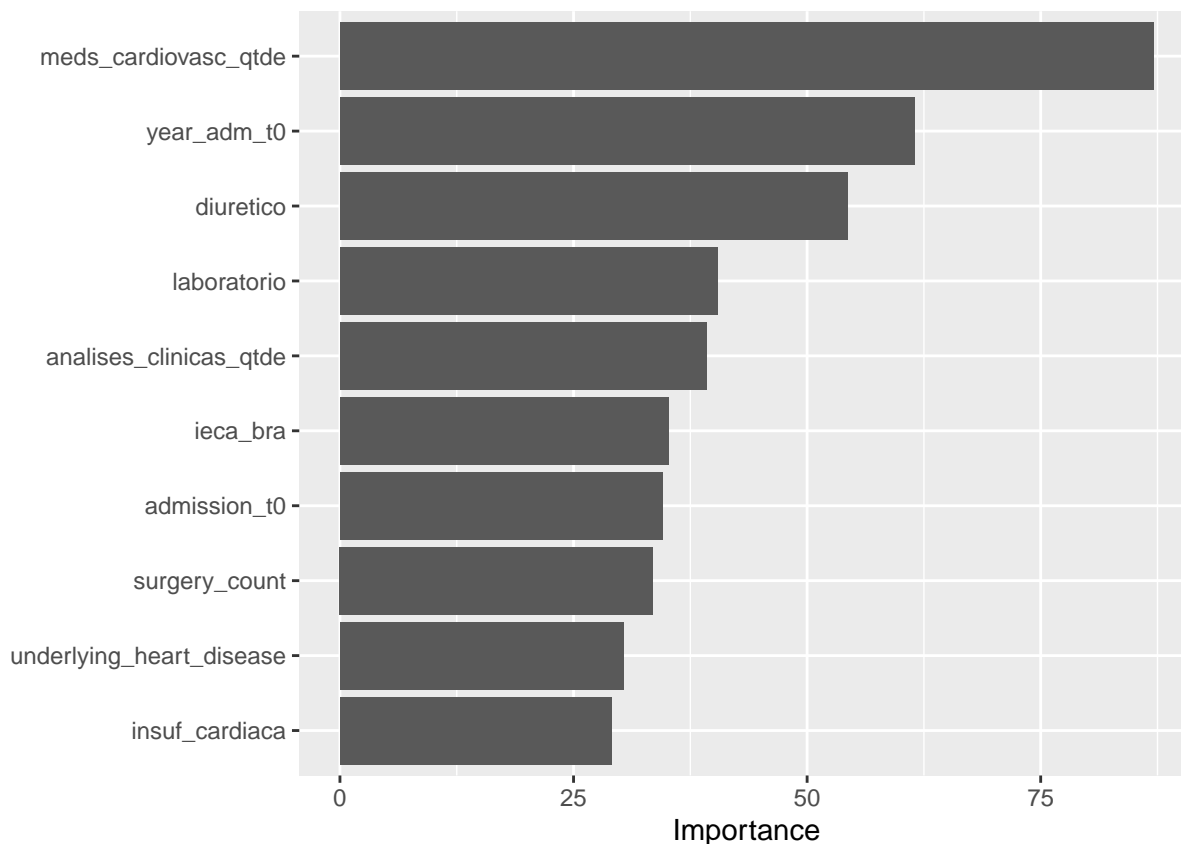
```



```

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",
    "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()) %>%
  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)

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

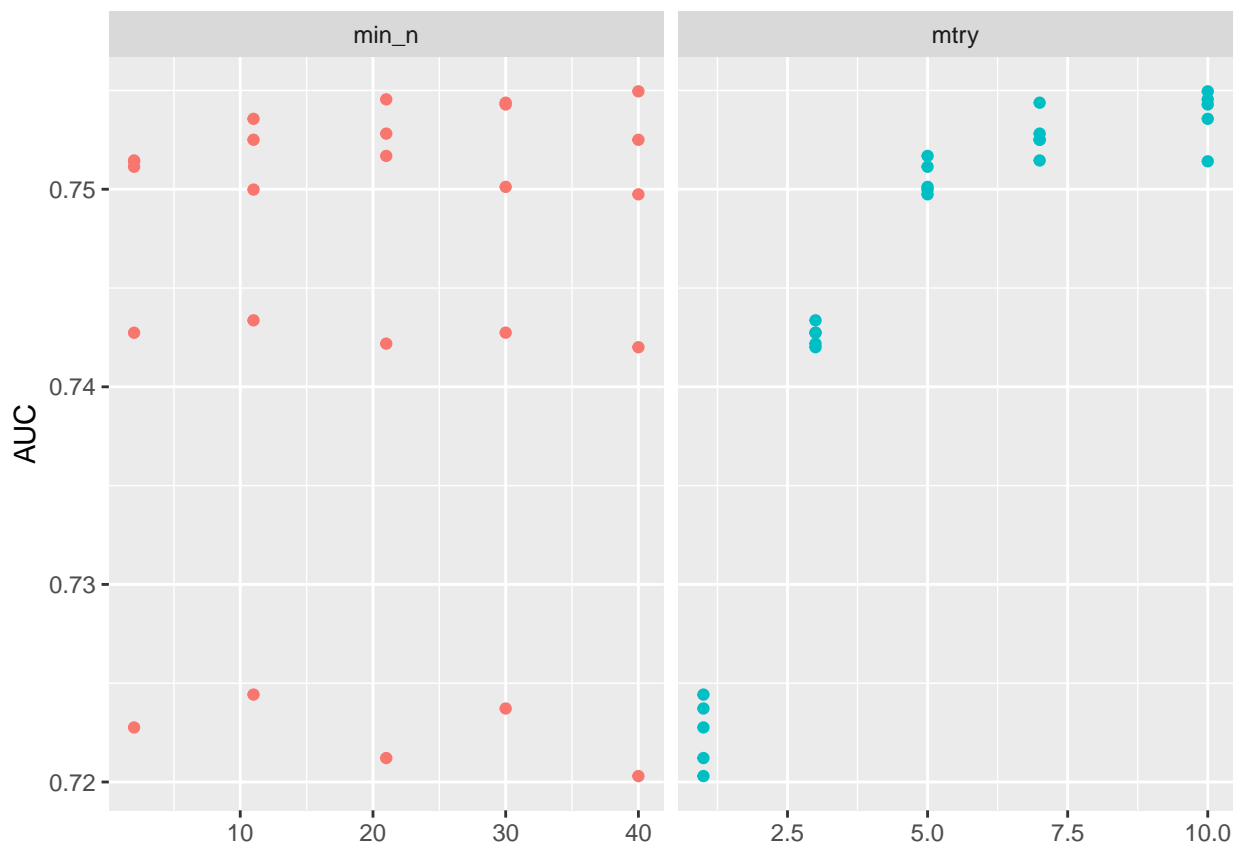
```
rf_tune <-  
  rf_workflow %>%  
    tune_grid(resamples = df_folds,  
              grid = rf_grid)
```

[illegible]

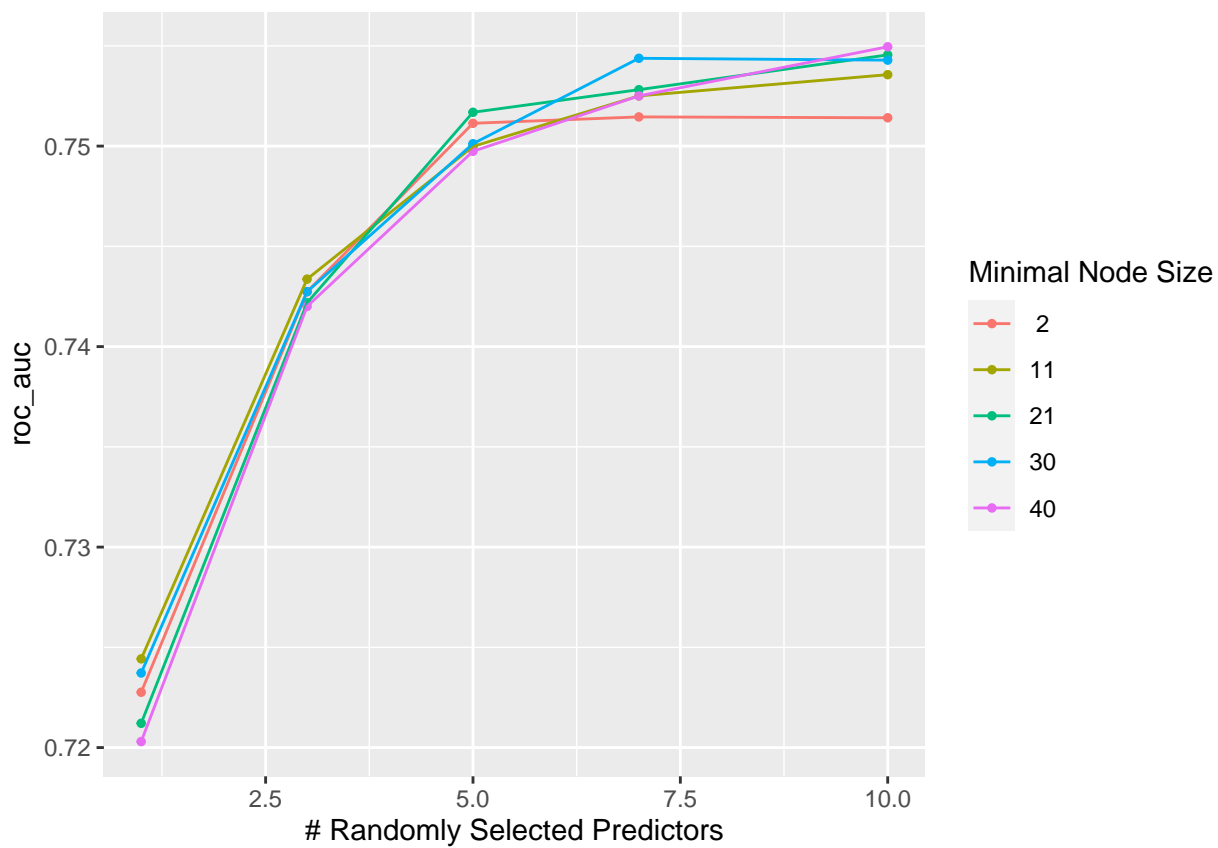

```
## ! Fold4: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 1/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 2/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 3/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 4/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 5/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 6/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 7/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 8/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 9/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 10/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 11/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 12/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 13/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 14/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 15/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 16/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 17/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 18/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 19/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 20/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 21/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 22/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 23/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 24/25 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 25/25 (predictions): There are new levels in a factor: NA
```

```
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)

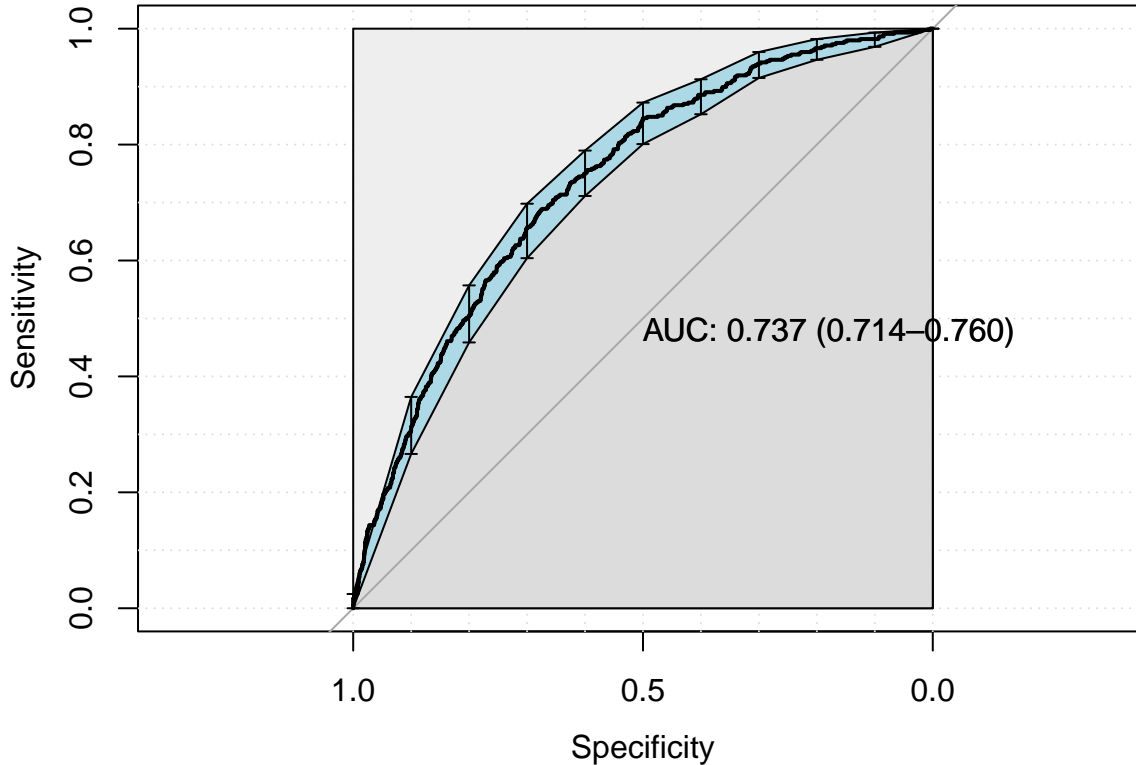
last_rf_fit <-
  final_rf_workflow %>%
  last_fit(df_split)

## ! train/test split: preprocessor 1/1: There are new levels in a factor: NA
## ! train/test split: preprocessor 1/1, model 1/1 (predictions): There are new levels in a factor: NA
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()

```

KNN

```

knn_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()) %>%
step_dummy(all_nominal_predictors()) %>%
step_impute_mode(all_nominal_predictors()) %>%
step_impute_mean(all_numeric_predictors())

knn_spec <-
  nearest_neighbor(neighbors = tune(),
                  # weight_func = tune(),
                  dist_power = tune()) %>%
  set_mode("classification") %>%
  set_engine("kkn")

knn_grid <- grid_regular(neighbors(),
                        # weight_func(),
                        dist_power(),
                        levels = 2)

knn_workflow <-
  workflow() %>%
  add_recipe(knn_recipe) %>%
  add_model(knn_spec)

knn_tune <-
  knn_workflow %>%
  tune_grid(resamples = df_folds,
            grid = knn_grid)

## ! Fold1: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold1: preprocessor 1/1, model 1/2 (predictions): There are new levels in a factor: NA
## ! Fold1: preprocessor 1/1, model 2/2 (predictions): There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1, model 1/2 (predictions): There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1, model 2/2 (predictions): There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1, model 1/2 (predictions): There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1, model 2/2 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 1/2 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 2/2 (predictions): There are new levels in a factor: NA

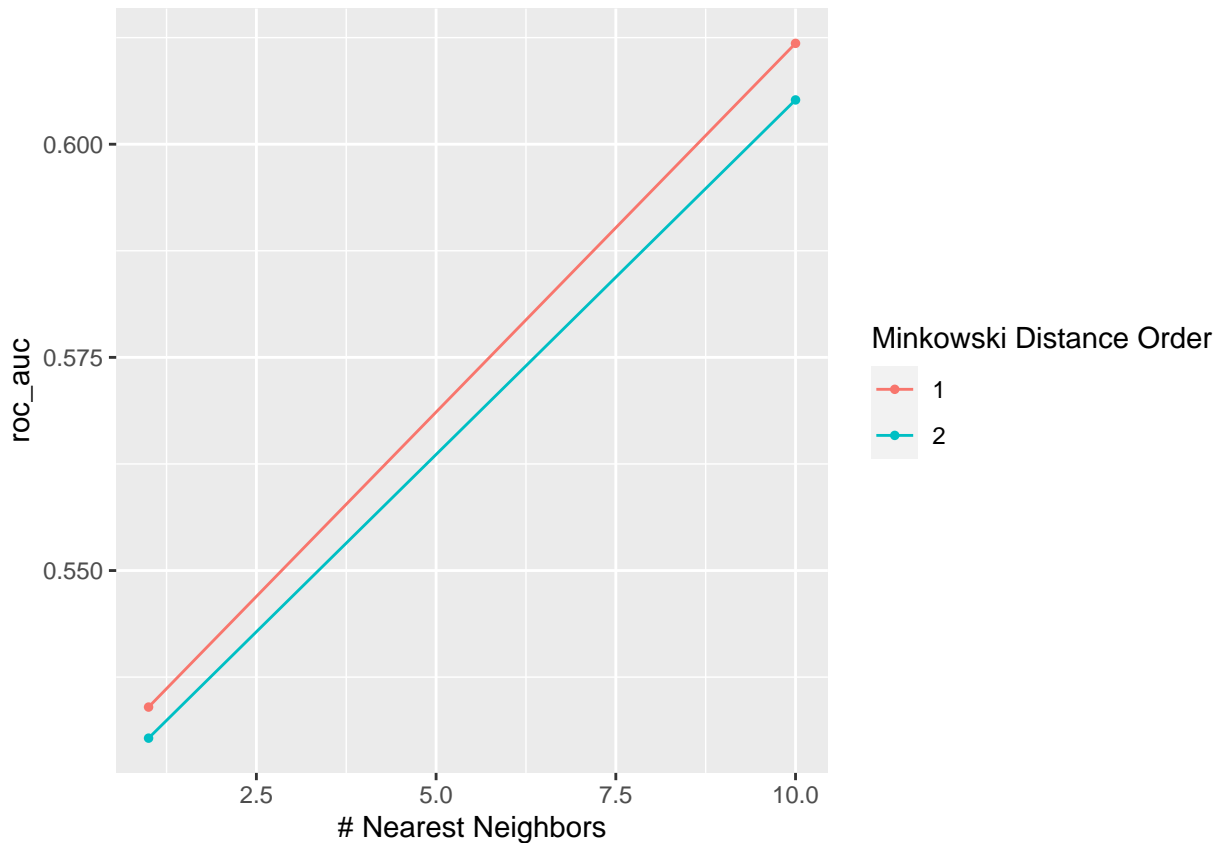
knn_tune %>%
  collect_metrics()

## # A tibble: 8 x 8
##   neighbors dist_power .metric .estimator mean      n std_err .config
##   <int>      <dbl> <chr>    <chr>    <dbl> <int>   <dbl> <chr>
## 1         1         1 accuracy binary    0.864     4 0.000824 Preprocessor1_Model1
## 2         1         1 roc_auc  binary    0.534     4 0.00517  Preprocessor1_Model1
## 3        10         1 accuracy binary    0.901     4 0.00201  Preprocessor1_Model2
## 4        10         1 roc_auc  binary    0.612     4 0.00883  Preprocessor1_Model2
## 5         1         2 accuracy binary    0.853     4 0.000521 Preprocessor1_Model3
## 6         1         2 roc_auc  binary    0.530     4 0.00432  Preprocessor1_Model3
## 7        10         2 accuracy binary    0.899     4 0.00146  Preprocessor1_Model4

```

```
## 8      10      2 roc_auc binary      0.605      4 0.00276 Preprocessor1_Model4
```

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



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

```
## # A tibble: 4 x 8
##   neighbors dist_power .metric .estimator mean      n std_err .config
##   <int>      <dbl> <chr>   <chr>    <dbl> <int>  <dbl> <chr>
## 1      10         1 roc_auc binary    0.612     4 0.00883 Preprocessor1_Model2
## 2      10         2 roc_auc binary    0.605     4 0.00276 Preprocessor1_Model4
## 3       1         1 roc_auc binary    0.534     4 0.00517 Preprocessor1_Model1
## 4       1         2 roc_auc binary    0.530     4 0.00432 Preprocessor1_Model3
```

```
best_knn <- knn_tune %>%
  select_best("roc_auc")
```

```
final_knn_workflow <-
  knn_workflow %>%
  finalize_workflow(best_knn)
```

```
last_knn_fit <-
  final_knn_workflow %>%
  last_fit(df_split)
```

```
## ! train/test split: preprocessor 1/1: There are new levels in a factor: NA
```

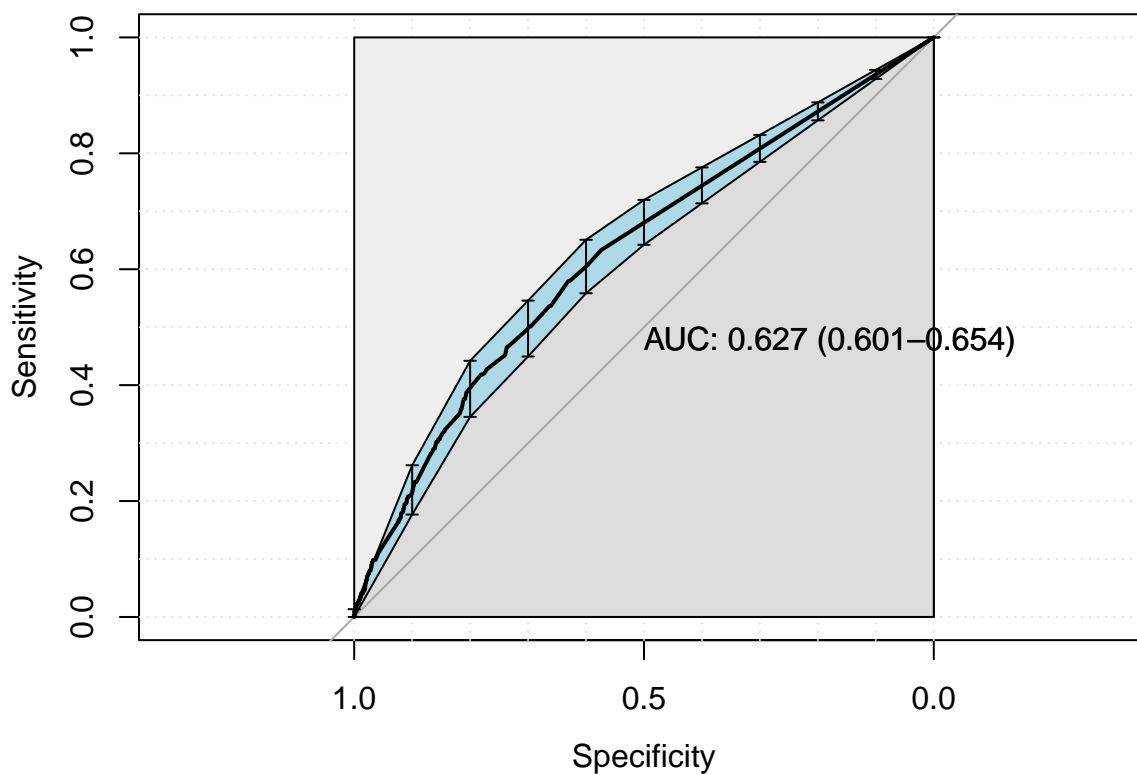
```
## ! train/test split: preprocessor 1/1, model 1/1 (predictions): There are new levels in a factor: NA
```

```
final_knn_fit <- extract_workflow(last_knn_fit)
```

```
knn_auc = validation(final_knn_fit, df_test)
```

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

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



```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                0 4231  430
##                1   52   17
##
##                Accuracy : 0.8981
##                95% CI : (0.8891, 0.9066)
##                No Information Rate : 0.9055
##                P-Value [Acc > NIR] : 0.96
##
##                Kappa : 0.0417
##
## McNemar's Test P-Value : <2e-16
##
##                Sensitivity : 0.98786
##                Specificity : 0.03803
##                Pos Pred Value : 0.90775
##                Neg Pred Value : 0.24638
##                Prevalence : 0.90550
##                Detection Rate : 0.89450
##                Detection Prevalence : 0.98541
##                Balanced Accuracy : 0.51295
##
##                'Positive' Class : 0
##
```

SVM

```
svm_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()) %>%
step_dummy(all_nominal_predictors()) %>%
step_impute_mode(all_nominal_predictors()) %>%
step_impute_mean(all_numeric_predictors())

svm_spec <-
  svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
  set_mode("classification") %>%
  set_engine("kernlab")

svm_grid <- grid_regular(cost(),
  rbf_sigma(),
  levels = 2)

svm_workflow <-
  workflow() %>%
  add_recipe(svm_recipe) %>%
  add_model(svm_spec)

svm_tune <-
  svm_workflow %>%
  tune_grid(resamples = df_folds,
    grid = svm_grid)

## ! Fold1: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold1: preprocessor 1/1, model 1/4: Variable(s) '' constant. Cannot scale data.
## ! Fold1: preprocessor 1/1, model 1/4 (predictions): There are new levels in a factor: NA
## ! Fold1: preprocessor 1/1, model 2/4: Variable(s) '' constant. Cannot scale data.
## ! Fold1: preprocessor 1/1, model 2/4 (predictions): There are new levels in a factor: NA
## ! Fold1: preprocessor 1/1, model 3/4: Variable(s) '' constant. Cannot scale data.
## ! Fold1: preprocessor 1/1, model 3/4 (predictions): There are new levels in a factor: NA
## ! Fold1: preprocessor 1/1, model 4/4: Variable(s) '' constant. Cannot scale data.
## ! Fold1: preprocessor 1/1, model 4/4 (predictions): There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1, model 1/4: Variable(s) '' constant. Cannot scale data.
## ! Fold2: preprocessor 1/1, model 1/4 (predictions): There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1, model 2/4: Variable(s) '' constant. Cannot scale data.
## ! Fold2: preprocessor 1/1, model 2/4 (predictions): There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1, model 3/4: Variable(s) '' constant. Cannot scale data.
## ! Fold2: preprocessor 1/1, model 3/4 (predictions): There are new levels in a factor: NA
## ! Fold2: preprocessor 1/1, model 4/4: Variable(s) '' constant. Cannot scale data.
## ! Fold2: preprocessor 1/1, model 4/4 (predictions): There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1, model 1/4: Variable(s) '' constant. Cannot scale data.

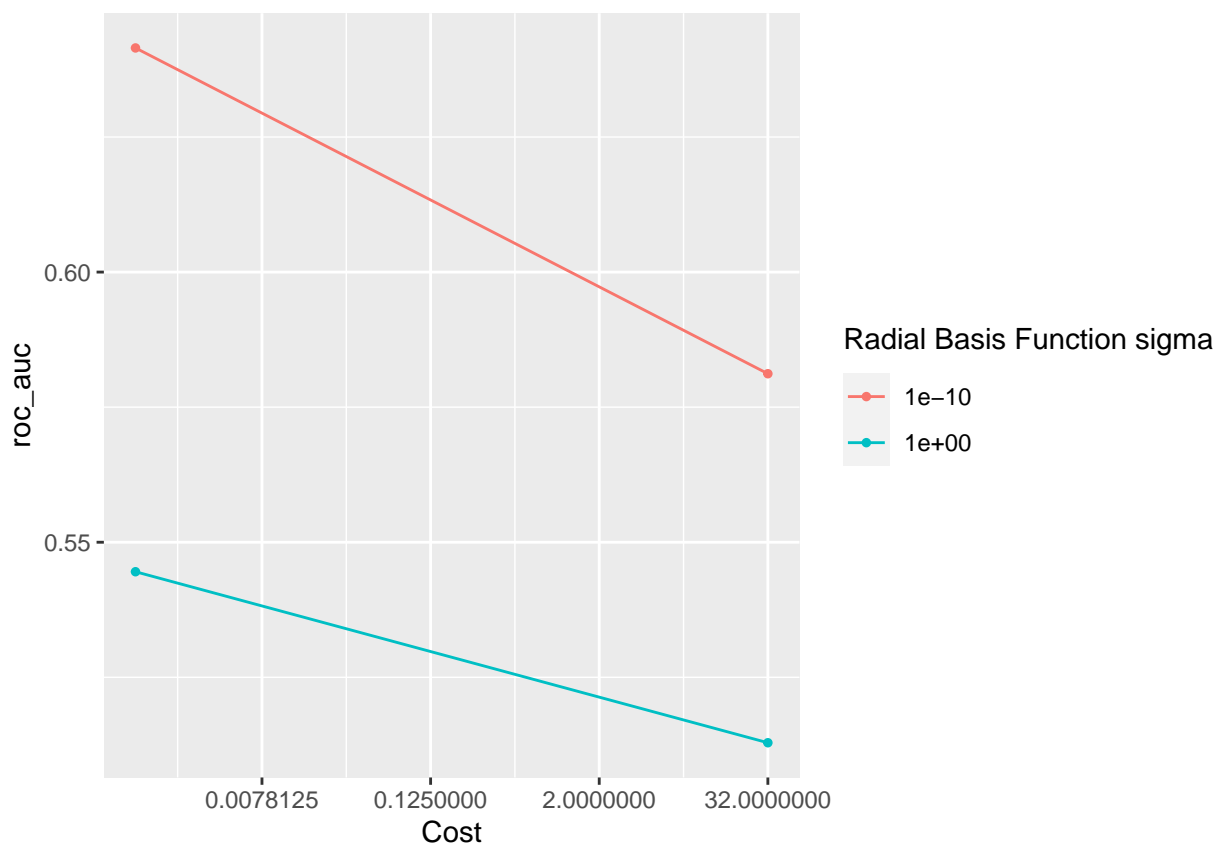
```

```
## ! Fold3: preprocessor 1/1, model 1/4 (predictions): There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1, model 2/4: Variable(s) '' constant. Cannot scale data.
## ! Fold3: preprocessor 1/1, model 2/4 (predictions): There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1, model 3/4: Variable(s) '' constant. Cannot scale data.
## ! Fold3: preprocessor 1/1, model 3/4 (predictions): There are new levels in a factor: NA
## ! Fold3: preprocessor 1/1, model 4/4: Variable(s) '' constant. Cannot scale data.
## ! Fold3: preprocessor 1/1, model 4/4 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1: There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 1/4: Variable(s) '' constant. Cannot scale data.
## ! Fold4: preprocessor 1/1, model 1/4 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 2/4: Variable(s) '' constant. Cannot scale data.
## ! Fold4: preprocessor 1/1, model 2/4 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 3/4: Variable(s) '' constant. Cannot scale data.
## ! Fold4: preprocessor 1/1, model 3/4 (predictions): There are new levels in a factor: NA
## ! Fold4: preprocessor 1/1, model 4/4: Variable(s) '' constant. Cannot scale data.
## ! Fold4: preprocessor 1/1, model 4/4 (predictions): There are new levels in a factor: NA
```

```
svm_tune %>%
  collect_metrics()
```

```
## # A tibble: 8 x 8
##       cost    rbf_sigma .metric .estimator mean    n std_err .config
##       <dbl>      <dbl> <chr>   <chr>      <dbl> <int>  <dbl> <chr>
## 1  0.000977 0.0000000001 accuracy binary    0.906     4 0.000869 Preprocessor1_Model1
## 2  0.000977 0.0000000001 roc_auc  binary    0.641     4 0.00725  Preprocessor1_Model1
## 3  32        0.0000000001 accuracy binary    0.906     4 0.000869 Preprocessor1_Model2
## 4  32        0.0000000001 roc_auc  binary    0.581     4 0.0385   Preprocessor1_Model2
## 5  0.000977 1          accuracy binary    0.906     4 0.000869 Preprocessor1_Model3
## 6  0.000977 1          roc_auc  binary    0.545     4 0.00785  Preprocessor1_Model3
## 7  32        1          accuracy binary    0.905     4 0.000942 Preprocessor1_Model4
## 8  32        1          roc_auc  binary    0.513     4 0.0366   Preprocessor1_Model4
```

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

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

```
## # A tibble: 4 x 8
##   cost    rbf_sigma .metric .estimator mean    n std_err .config
##   <dbl>    <dbl> <chr>   <chr>   <dbl> <int>  <dbl> <chr>
## 1 0.000977 0.0000000001 roc_auc binary 0.641     4 0.00725 Preprocessor1_Model1
## 2 32      0.0000000001 roc_auc binary 0.581     4 0.0385  Preprocessor1_Model2
## 3 0.000977 1          roc_auc binary 0.545     4 0.00785 Preprocessor1_Model3
## 4 32      1          roc_auc binary 0.513     4 0.0366  Preprocessor1_Model4
```

```
best_svm <- svm_tune %>%
  select_best("roc_auc")
```

```
final_svm_workflow <-
  svm_workflow %>%
  finalize_workflow(best_svm)
```

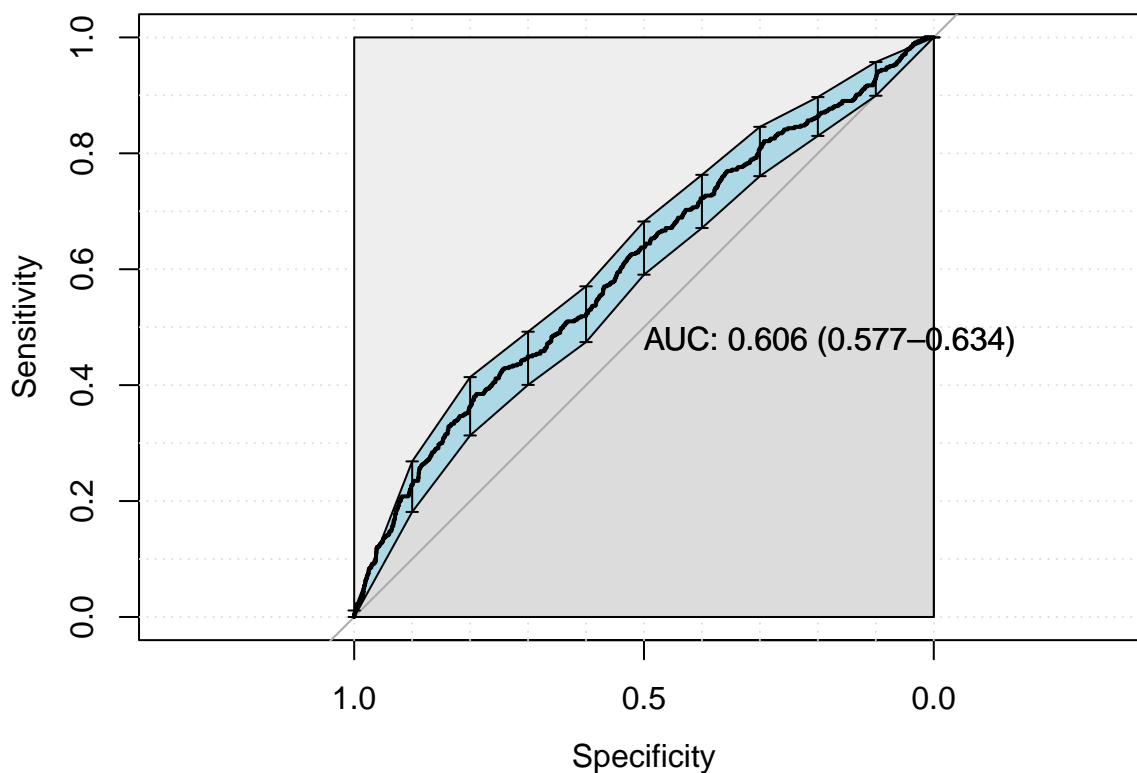
```
last_svm_fit <-
  final_svm_workflow %>%
  last_fit(df_split)
```

```
## ! train/test split: preprocessor 1/1: There are new levels in a factor: NA
## ! train/test split: preprocessor 1/1, model 1/1: Variable(s) ' ' constant. Cannot scale data.
## ! train/test split: preprocessor 1/1, model 1/1 (predictions): There are new levels in a factor: NA
final_svm_fit <- extract_workflow(last_svm_fit)
```

```
svm_auc = validation(final_svm_fit, df_test)
```

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

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



```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                0 4283  447
##                1    0    0
##
##          Accuracy : 0.9055
##          95% CI   : (0.8968, 0.9137)
##    No Information Rate : 0.9055
##    P-Value [Acc > NIR] : 0.5126
##
##          Kappa : 0
##
##  McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.0000
##          Specificity : 0.0000
##    Pos Pred Value : 0.9055
##    Neg Pred Value :   NaN
##          Prevalence : 0.9055
##    Detection Rate : 0.9055
##    Detection Prevalence : 1.0000
##    Balanced Accuracy : 0.5000
##
##          'Positive' Class : 0
##
```

Models Comparison

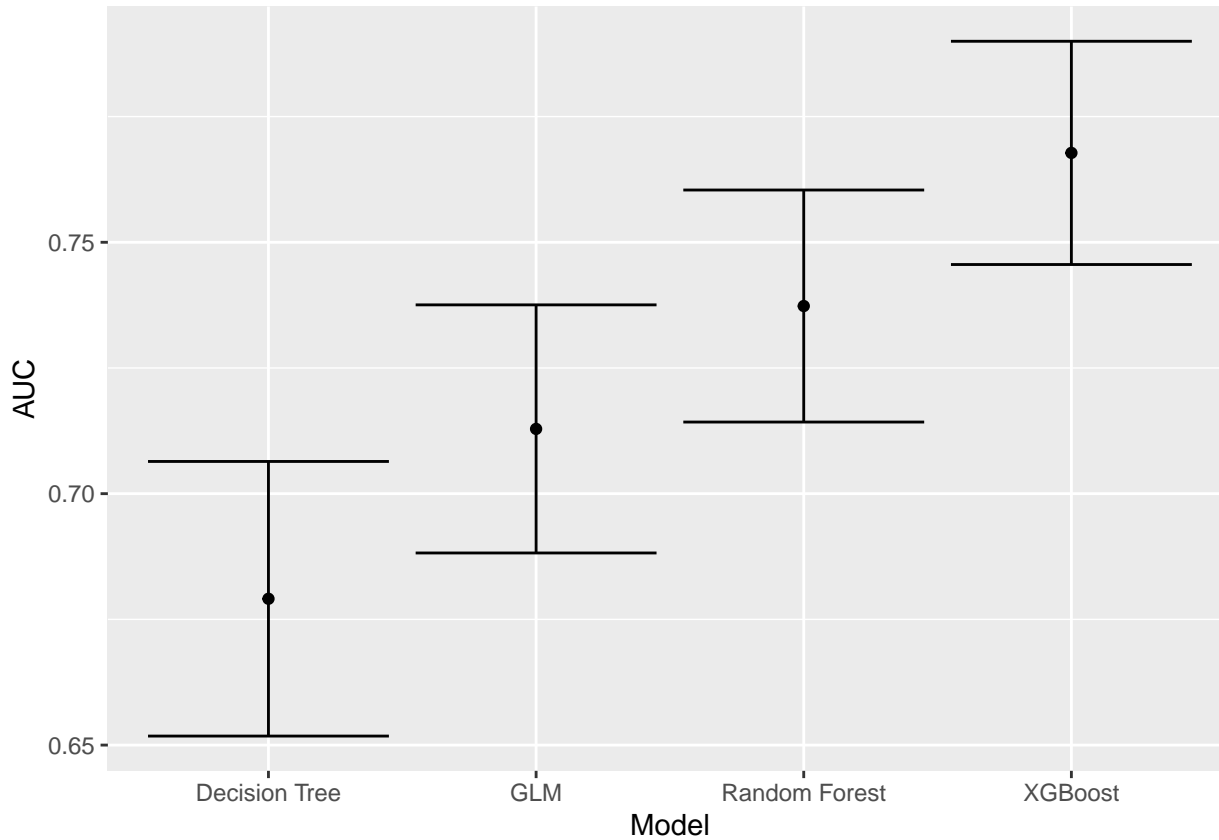
```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
```

```

'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))

```