

Model Selection - death_2year

Eduardo Yuki Yada

Global parameters

```
k <- params$k # Number of folds for cross validation
grid_size <- params$grid_size # Number of parameter combination to tune on each model
repeats <- params$repeats
RUN_ALL_MODELS <- params$RUN_ALL_MODELS
Hmisc::list.tree(params)
```

```
## params = list 5 (952 bytes)
## . outcome_column = character 1= death_2year
## . k = double 1= 10
## . grid_size = double 1= 20
## . repeats = double 1= 2
## . RUN_ALL_MODELS = logical 1= TRUE
```

Minutes to run: 0

Imports

```
library(tidyverse)
library(yaml)
library(tidymodels)
library(usemodels)
library(vip)
library(bonsai)
library(lightgbm)
library(caret)
library(pROC)

source("aux_functions.R")
predict <- stats::predict
```

Minutes to run: 0

Loading data

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

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

outcome_column <- params$outcome_column
features_list <- params$features_list

df <- mutate(df, across(where(is.character), as.factor))
```

Minutes to run: 0.006

```
dir.create(file.path("./auxiliar/model_selection/hyperparameters/"),
           showWarnings = FALSE,
```

```

        recursive = TRUE)

dir.create(file.path("./auxiliar/model_selection/performance/"),
          showWarnings = FALSE,
          recursive = TRUE)

```

Minutes to run: 0

Eligible features

```

cat_features_list = read_yaml(sprintf(
  "./auxiliar/significant_columns/categorical_%s.yaml",
  outcome_column
))

num_features_list = read_yaml(sprintf(
  "./auxiliar/significant_columns/numerical_%s.yaml",
  outcome_column
))

features_list = c(cat_features_list, num_features_list)

```

Minutes to run: 0

```

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

exception_columns = c('death_intraop', 'death_intraop_1', 'disch_outcomes_t0')

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

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

features = base::intersect(eligible_features, features_list)

gluedown::md_order(features, seq = TRUE, pad = TRUE)

## 01. sex
## 02. age
## 03. education_level
## 04. underlying_heart_disease
## 05. heart_disease
## 06. nyha_basal
## 07. hypertension
## 08. prior_mi
## 09. heart_failure
## 10. af
## 11. cardiac_arrest
## 12. valvopathy
## 13. diabetes

```

14. renal_failure
15. hemodialysis
16. stroke
17. copd
18. comorbidities_count
19. procedure_type_1
20. reop_type_1
21. procedure_type_new
22. cied_final_1
23. cied_final_group_1
24. admission_pre_t0_count
25. admission_pre_t0_180d
26. year_adm_t0
27. icu_t0
28. dialysis_t0
29. admission_t0_emergency
30. aco
31. antiarritmico
32. ieca_bra
33. dva
34. digoxina
35. estatina
36. diuretico
37. vasodilatador
38. insuf_cardiaca
39. espirolactona
40. antiplaquetario_ev
41. insulina
42. psicofarmacos
43. antifungico
44. antiviral
45. classe_meds_qtde
46. meds_cardiovasc_qtde
47. meds_antimicrobianos
48. vni
49. ventilacao_mecanica
50. transplante_cardiaco
51. outros_proced_cirurgicos
52. icp
53. angioplastia
54. cateterismo
55. cateter_venoso_central
56. proced_invasivos_qtde
57. transfusao
58. interconsulta
59. equipe_multiprof
60. holter
61. teste_esforco
62. tilt_teste
63. metodos_graficos_qtde
64. laboratorio
65. cultura
66. analises_clinicas_qtde
67. citologia
68. histopatologia_qtde
69. angio_tc
70. cintilografia
71. ecocardiograma
72. endoscopia
73. flebografia
74. pet_ct

```
## 75. ultrassom
## 76. tomografia
## 77. ressonancia
## 78. exames_imagem_qtde
## 79. bic
## 80. hospital_stay
```

Minutes to run: 0

Train test split (70%/30%)

```
set.seed(42)

if (outcome_column == 'readmission_30d') {
  df_split <- readRDS("./dataset/split_object.rds")
} else {
  df_split <- initial_split(df, prop = .7, strata = all_of(outcome_column))
}

df_train <- training(df_split) %>% dplyr::select(all_of(c(features, outcome_column)))
df_test <- testing(df_split) %>% dplyr::select(all_of(c(features, outcome_column)))

df_folds <- vfold_cv(df_train, v = k,
                     strata = all_of(outcome_column))
```

Minutes to run: 0.001

Boosted Tree (XGBoost)

```
xgboost_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors())

xgboost_spec <- boost_tree(
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
) %>%
  set_engine("xgboost",
             nthread = 8) %>%
  set_mode("classification")

xgboost_grid <- grid_latin_hypercube(
  trees(range = c(25L, 150L)),
  min_n(range = c(2L, 100L)),
  tree_depth(range = c(2L, 15L)),
  learn_rate(range = c(-3, -1), trans = log10_trans()),
  size = grid_size
)

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

xgboost_tune <-
```

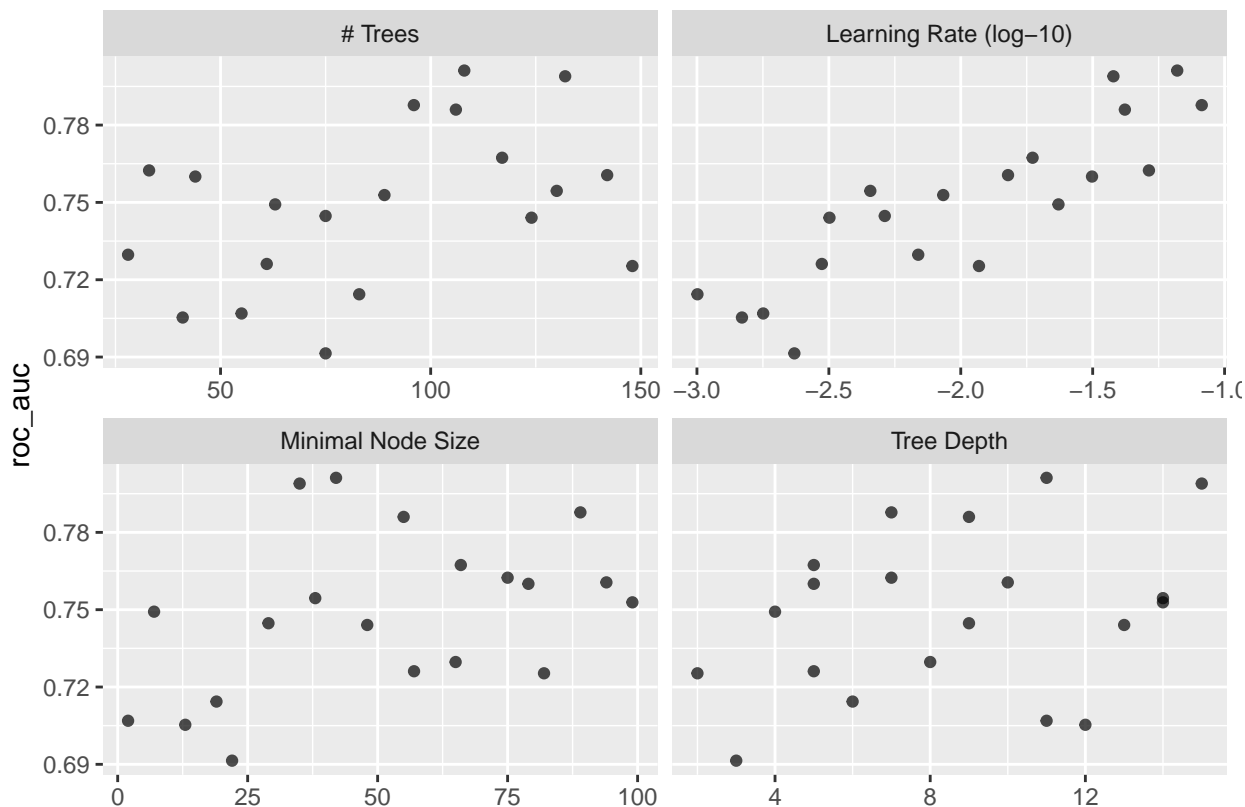
```
xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)

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

```
## # A tibble: 5 x 10
##   trees min_n tree_depth learn_rate .metric .estimator mean      n std_err .config
##   <int> <int>    <int>    <dbl> <chr>   <chr>   <dbl> <int>  <dbl> <chr>
## 1   108    42      11    0.0661 roc_auc binary  0.801    10 0.00857 Preprocessor1_Model109
## 2   132    35      15    0.0379 roc_auc binary  0.799    10 0.00895 Preprocessor1_Model107
## 3    96    89       7    0.0819 roc_auc binary  0.788    10 0.00868 Preprocessor1_Model118
## 4   106    55       9    0.0419 roc_auc binary  0.786    10 0.00821 Preprocessor1_Model111
## 5   117    66       5    0.0187 roc_auc binary  0.767    10 0.00772 Preprocessor1_Model114
```

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

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

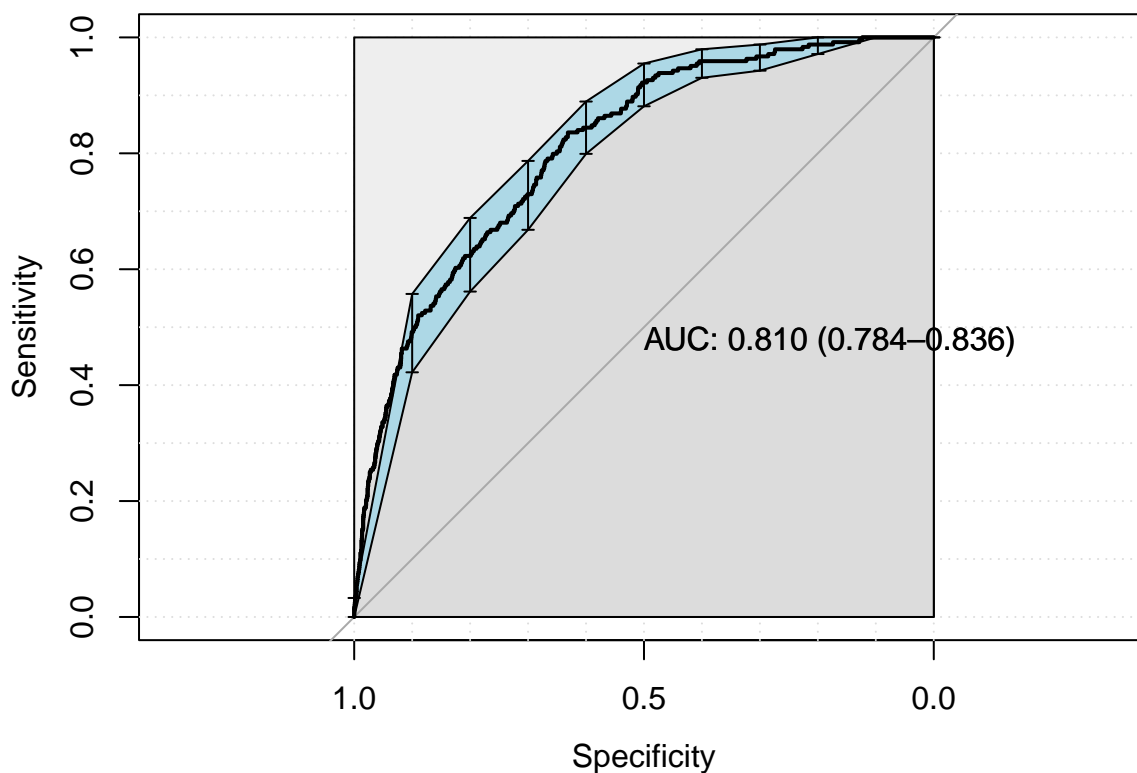


```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

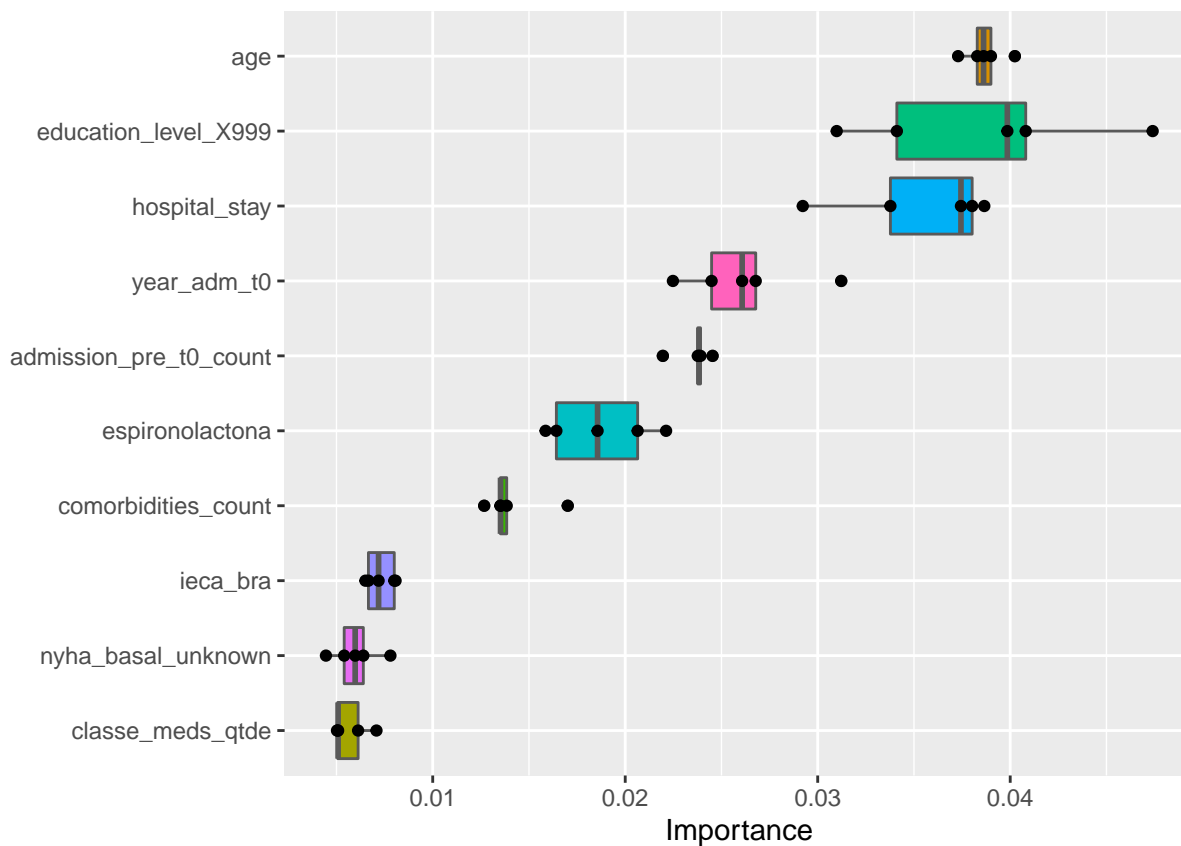
last_xgboost_fit <-
  final_xgboost_workflow %>%
  last_fit(df_split)

final_xgboost_fit <- extract_workflow(last_xgboost_fit)

xgboost_auc <- validation(final_xgboost_fit, df_test)
```



```
## [1] "Optimal Threshold: 0.03"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 2832   40
##    1 1654  204
##
##              Accuracy : 0.6419
##              95% CI   : (0.628, 0.6555)
##    No Information Rate : 0.9484
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1132
##
## Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6313
##              Specificity : 0.8361
##              Pos Pred Value : 0.9861
##              Neg Pred Value : 0.1098
##              Prevalence : 0.9484
##              Detection Rate : 0.5987
##              Detection Prevalence : 0.6072
##              Balanced Accuracy : 0.7337
##
##              'Positive' Class : 0
##
extract_vip(final_xgboost_fit, pred_wrapper = predict,
            reference_class = "0")
```



```
xgboost_parameters <- xgboost_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(-.metric, -.estimator, -.config, -mean, -n, -std_err) %>%
  as.list
```

Minutes to run: 4.474

Boosted Tree (LightGBM)

```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_nominal(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors())

lightgbm_spec <- boost_tree(
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  sample_size = 1
) %>%
  set_engine("lightgbm",
    nthread = 8) %>%
  set_mode("classification")

lightgbm_grid <- grid_latin_hypercube(
  trees(range = c(25L, 150L)),
  min_n(range = c(2L, 100L)),
  tree_depth(range = c(2L, 15L)),
  learn_rate(range = c(-3, -1), trans = log10_trans()),
  size = grid_size
```

```
)
```

```
lightgbm_workflow <-  
  workflow() %>%  
  add_recipe(lightgbm_recipe) %>%  
  add_model(lightgbm_spec)
```

```
lightgbm_tune <-  
  lightgbm_workflow %>%  
  tune_grid(resamples = df_folds,  
            grid = lightgbm_grid)
```

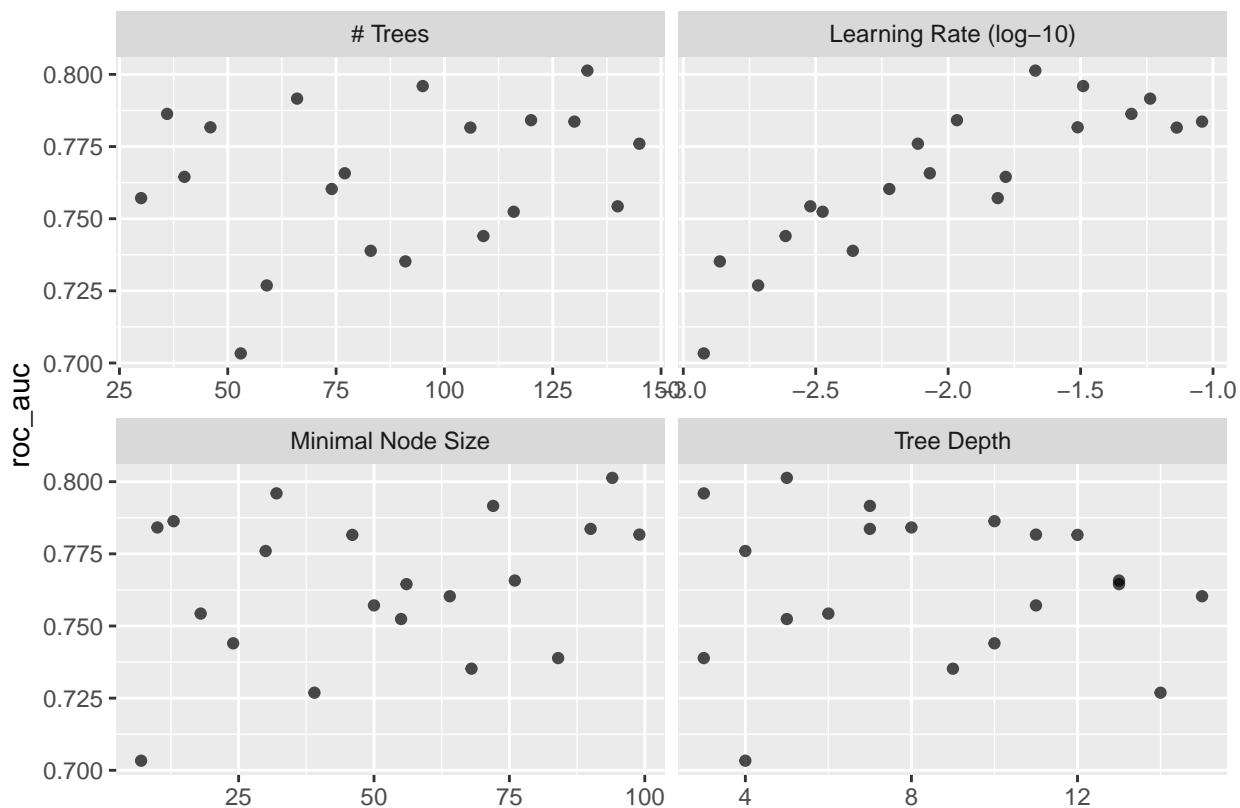
```
lightgbm_tune %>%  
  show_best("roc_auc")
```

```
## # A tibble: 5 x 10
```

```
##   trees min_n tree_depth learn_rate .metric .estimator mean      n std_err .config  
##   <int> <int>    <int>    <dbl> <chr>   <chr>   <dbl> <int>  <dbl> <chr>  
## 1   133    94        5    0.0213 roc_auc binary  0.801    10 0.00850 Preprocessor1_Model19  
## 2    95    32        3    0.0323 roc_auc binary  0.796    10 0.00943 Preprocessor1_Model07  
## 3    66    72        7    0.0579 roc_auc binary  0.792    10 0.00873 Preprocessor1_Model15  
## 4    36    13       10    0.0492 roc_auc binary  0.786    10 0.00775 Preprocessor1_Model03  
## 5   120    10        8    0.0108 roc_auc binary  0.784    10 0.00910 Preprocessor1_Model02
```

```
best_lightgbm <- lightgbm_tune %>%  
  select_best("roc_auc")
```

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



```
final_lightgbm_workflow <-  
  lightgbm_workflow %>%  
  finalize_workflow(best_lightgbm)
```

```
last_lightgbm_fit <-
```



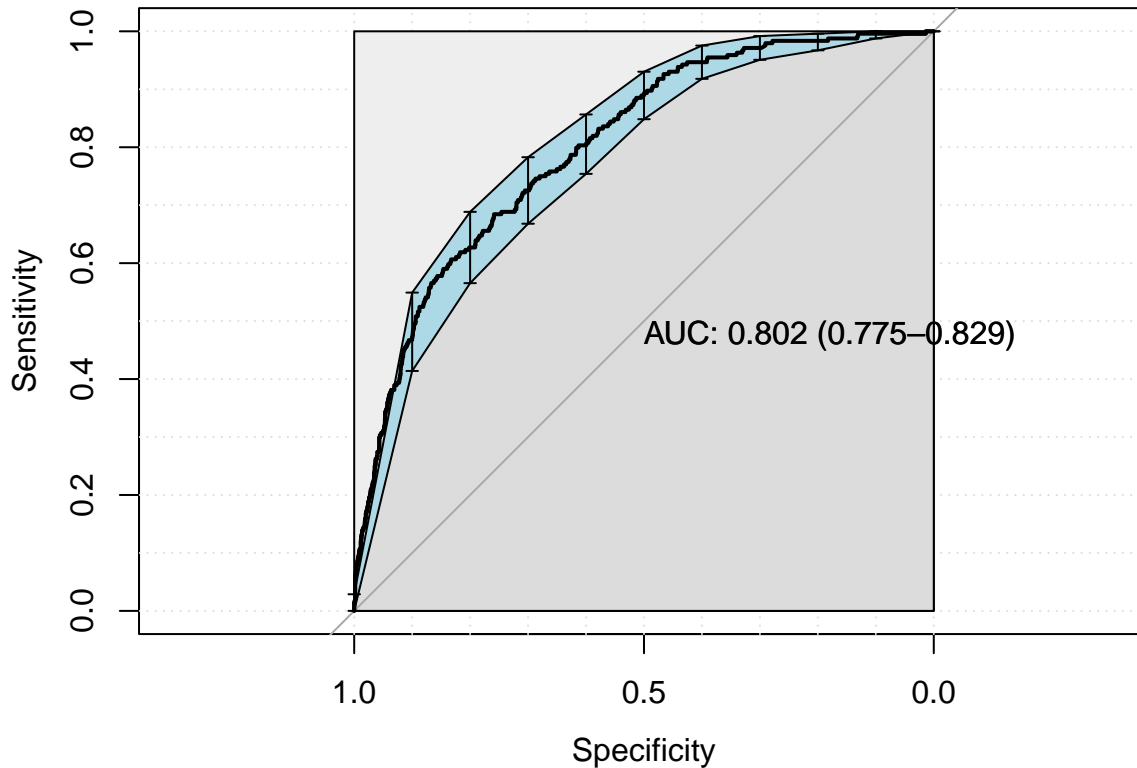
```

final_lightgbm_workflow %>%
  last_fit(df_split)

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_auc <- validation(final_lightgbm_fit, df_test)

```



```

## [1] "Optimal Threshold: 0.05"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 3404   77
## 1 1082  167
##
##              Accuracy : 0.755
##              95% CI   : (0.7425, 0.7672)
##    No Information Rate : 0.9484
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1504
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.7588
##              Specificity : 0.6844
##              Pos Pred Value : 0.9779
##              Neg Pred Value : 0.1337
##              Prevalence   : 0.9484
##              Detection Rate : 0.7197
##              Detection Prevalence : 0.7359
##              Balanced Accuracy : 0.7216
##

```

```
##      'Positive' Class : 0
##
lightgbm_parameters <- lightgbm_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(-.metric, -.estimator, -.config, -mean, -n, -std_err) %>%
  as.list

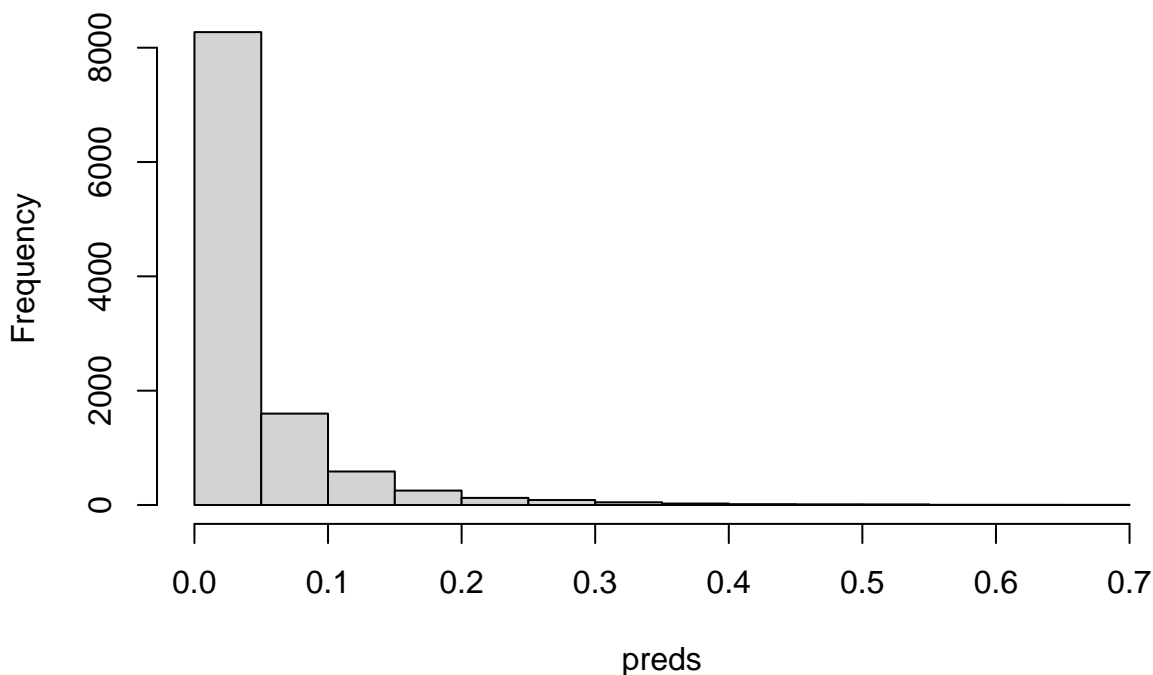
Hmisc::list.tree(lightgbm_parameters)

## lightgbm_parameters = list 4 (736 bytes)
## . trees = integer 1= 133
## . min_n = integer 1= 94
## . tree_depth = integer 1= 5
## . learn_rate = double 1= 0.02134

con <- file(sprintf('./auxiliar/model_selection/hyperparameters/%s.yaml', outcome_column), "w")
write_yaml(lightgbm_parameters, con)
close(con)
```

Minutes to run: 3.754

Histogram of preds



0.005

Minutes to run:

GLM

```
glmnet_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_numeric_predictors())
```

```

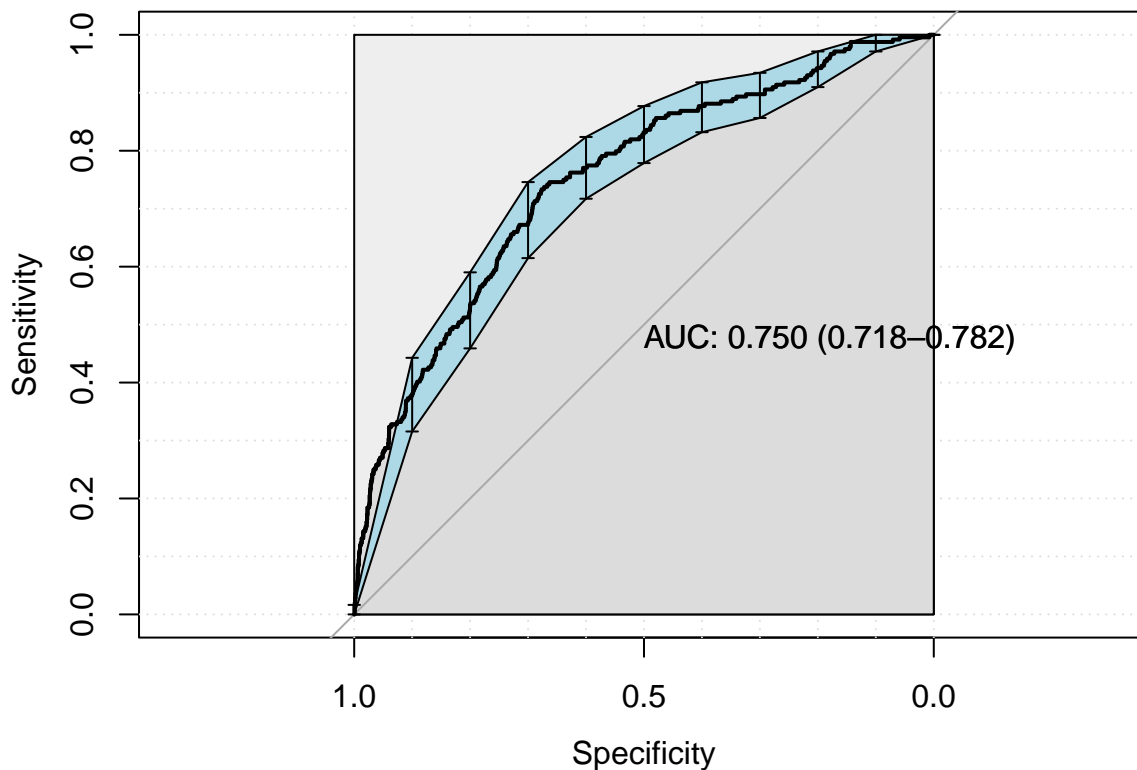
glmnet_spec <-
  logistic_reg(penalty = 0) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

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

glm_fit <- glmnet_workflow %>%
  fit(df_train)

glmnet_auc <- validation(glm_fit, df_test)

```



```

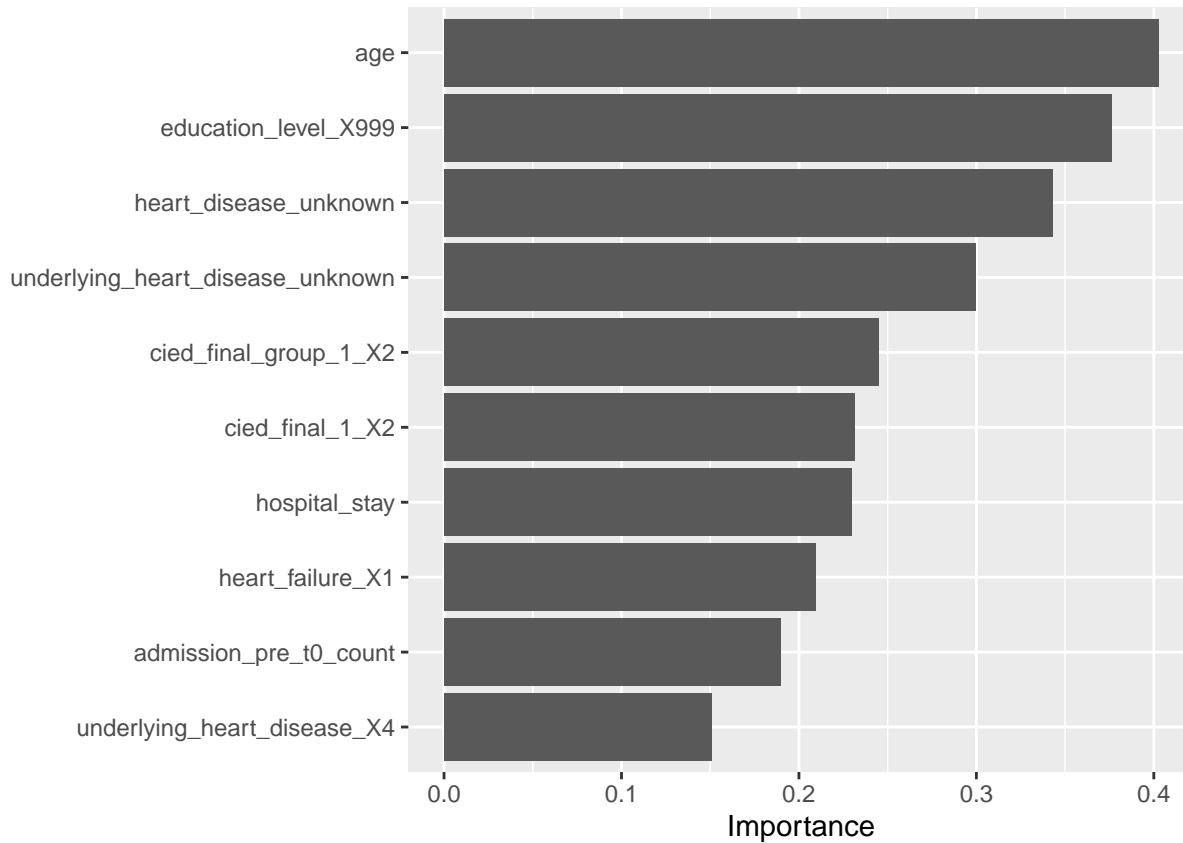
## [1] "Optimal Threshold: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 3036   65
## 1 1450  179
##
##              Accuracy : 0.6797
##              95% CI   : (0.6662, 0.693)
##    No Information Rate : 0.9484
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1114
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6768

```

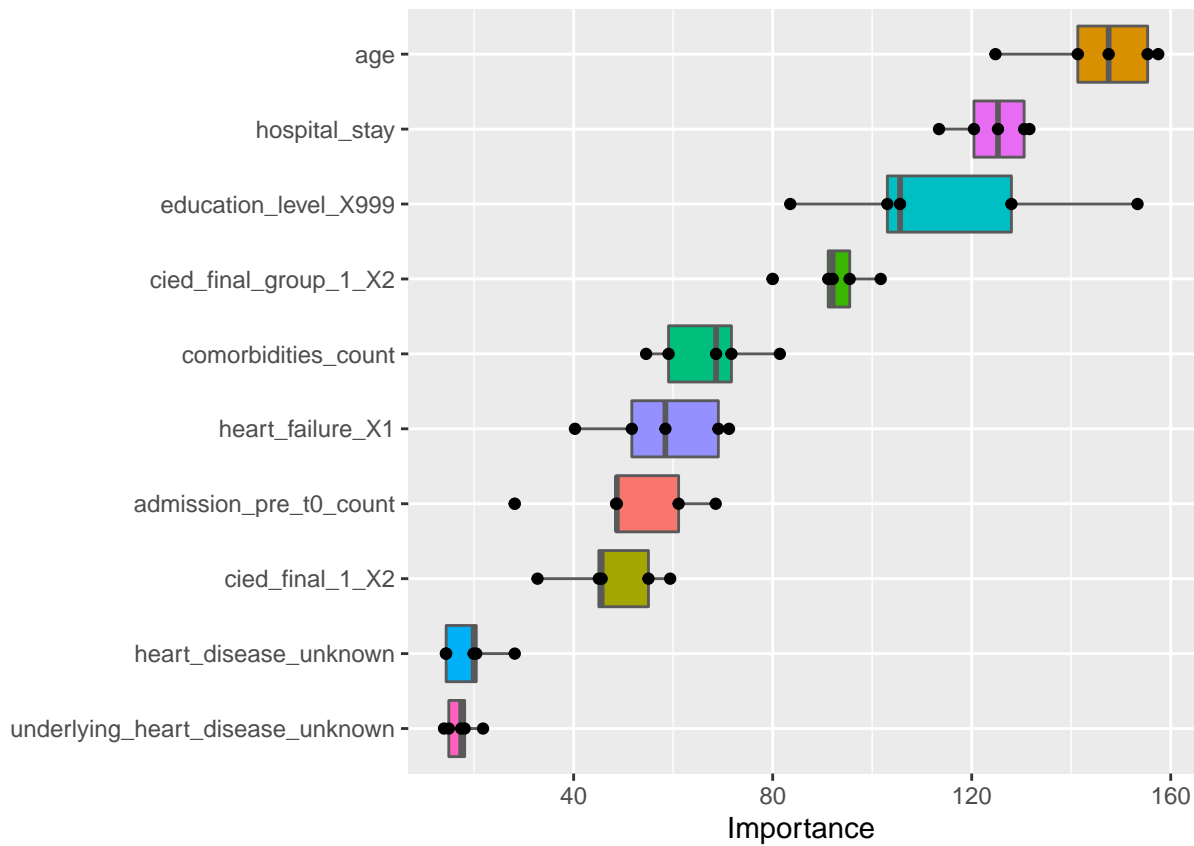
```
##          Specificity : 0.7336
##          Pos Pred Value : 0.9790
##          Neg Pred Value : 0.1099
##          Prevalence : 0.9484
##          Detection Rate : 0.6419
##          Detection Prevalence : 0.6556
##          Balanced Accuracy : 0.7052
##
##          'Positive' Class : 0
##
```

```
pfun_glmnet <- function(object, newdata) predict(object, newx = newdata)

extract_vip(glm_fit, pred_wrapper = pfun_glmnet,
            reference_class = "1", method = 'model')
```



```
extract_vip(glm_fit, pred_wrapper = pfun_glmnet,
            reference_class = "1", method = 'permute')
```



Minutes to run:

1.793

Decision Tree

```
tree_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

tree_spec <-
  decision_tree(cost_complexity = tune(),
                tree_depth = tune(),
                min_n = tune()) %>%
  set_mode("classification") %>%
  set_engine("rpart")

tree_grid <- grid_latin_hypercube(cost_complexity(),
                                   tree_depth(),
                                   min_n(),
                                   size = grid_size)

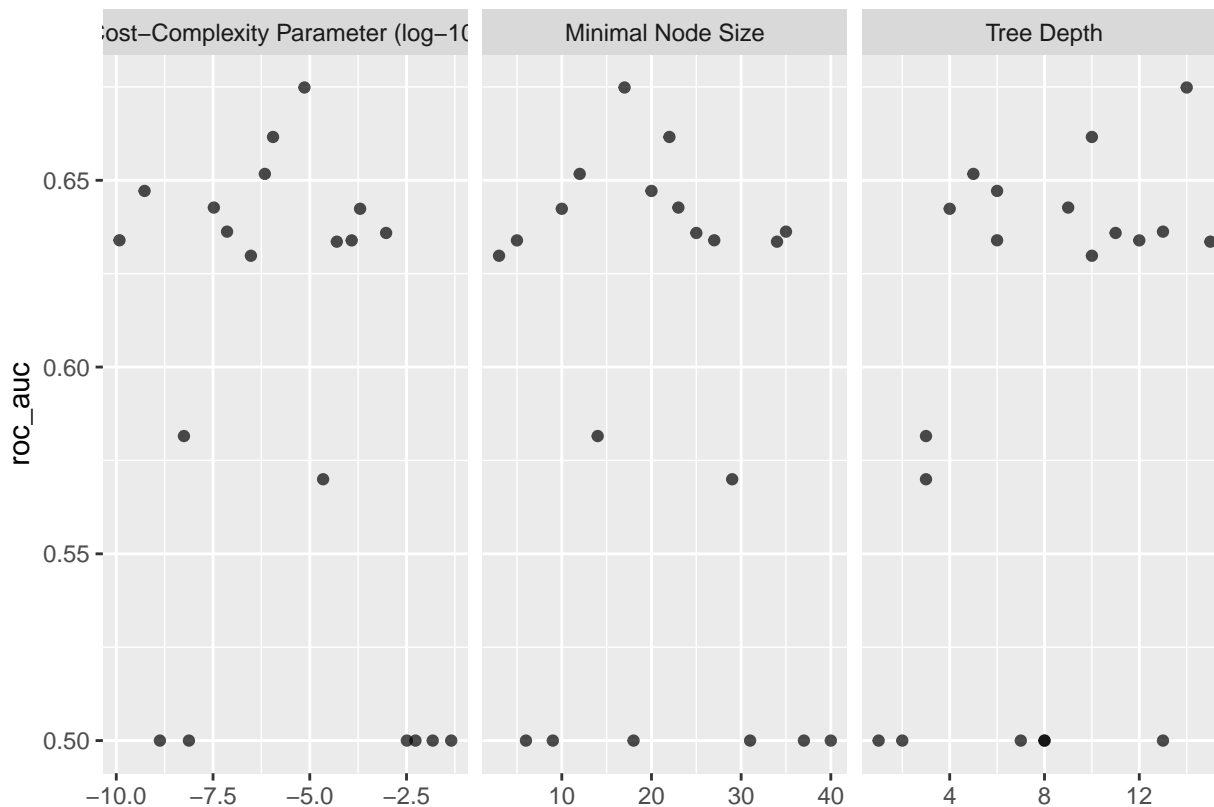
tree_workflow <-
  workflow() %>%
  add_recipe(tree_recipe) %>%
  add_model(tree_spec)

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

```
tree_tune %>%
  collect_metrics()
```

```
## # A tibble: 40 x 9
##   cost_complexity tree_depth min_n .metric .estimator mean      n std_err .config
##   <dbl>          <int> <int> <chr>   <chr>   <dbl> <int>   <dbl> <chr>
## 1      1.12e- 6         10     22 accuracy binary   0.947    10 0.00184 Preprocessor1_Model101
## 2      1.12e- 6         10     22 roc_auc  binary   0.662    10 0.0150  Preprocessor1_Model101
## 3      6.91e- 7          5     12 accuracy binary   0.951    10 0.00144 Preprocessor1_Model102
## 4      6.91e- 7          5     12 roc_auc  binary   0.652    10 0.0141  Preprocessor1_Model102
## 5      1.21e- 4         12      5 accuracy binary   0.938    10 0.00265 Preprocessor1_Model103
## 6      1.21e- 4         12      5 roc_auc  binary   0.634    10 0.0168  Preprocessor1_Model103
## 7      5.00e- 5         15     34 accuracy binary   0.950    10 0.00168 Preprocessor1_Model104
## 8      5.00e- 5         15     34 roc_auc  binary   0.634    10 0.00610 Preprocessor1_Model104
## 9      1.21e-10          6     27 accuracy binary   0.951    10 0.00165 Preprocessor1_Model105
## 10     1.21e-10          6     27 roc_auc  binary   0.634    10 0.00769 Preprocessor1_Model105
## # ... with 30 more rows
## # i Use 'print(n = ...)' to see more rows
```

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



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

```
## # A tibble: 5 x 9
##   cost_complexity tree_depth min_n .metric .estimator mean      n std_err .config
##   <dbl>          <int> <int> <chr>   <chr>   <dbl> <int>   <dbl> <chr>
## 1      7.30e- 6         14     17 roc_auc  binary   0.675    10 0.0112  Preprocessor1_Model114
## 2      1.12e- 6         10     22 roc_auc  binary   0.662    10 0.0150  Preprocessor1_Model101
## 3      6.91e- 7          5     12 roc_auc  binary   0.652    10 0.0141  Preprocessor1_Model102
## 4      5.38e-10          6     20 roc_auc  binary   0.647    10 0.0120  Preprocessor1_Model109
## 5      3.32e- 8          9     23 roc_auc  binary   0.643    10 0.00986 Preprocessor1_Model115
```

```

best_tree <- tree_tune %>%
  select_best("roc_auc")

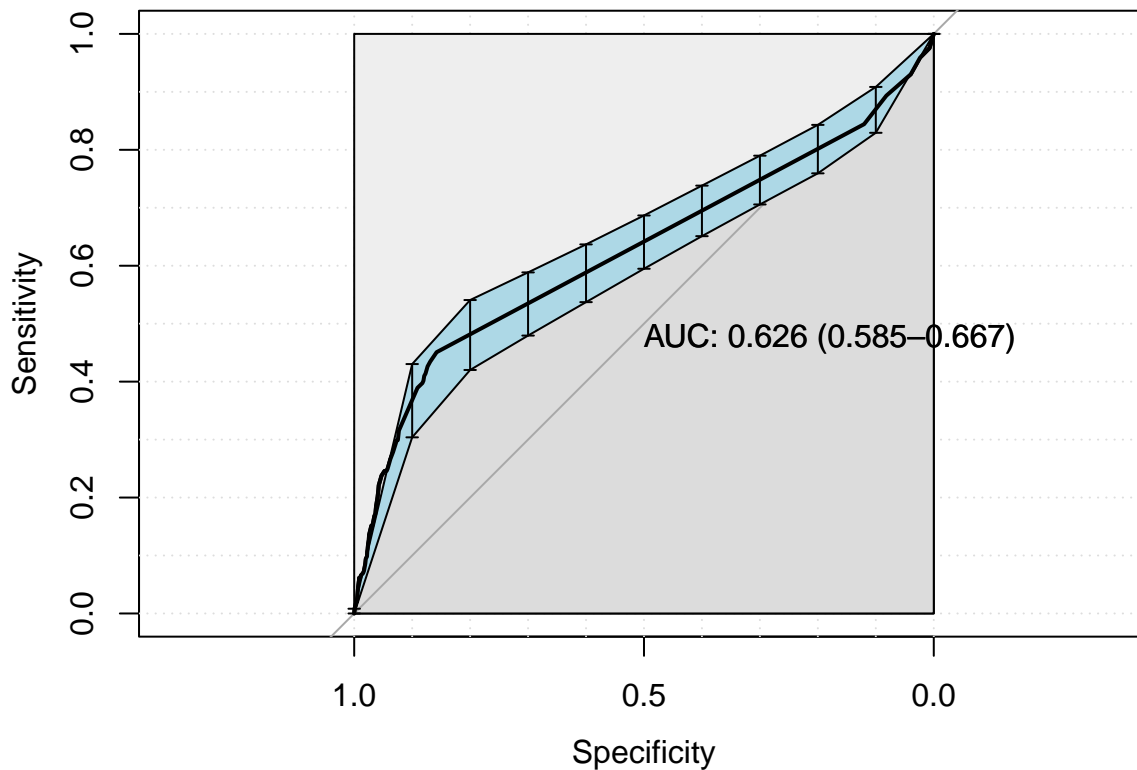
final_tree_workflow <-
  tree_workflow %>%
  finalize_workflow(best_tree)

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)

```



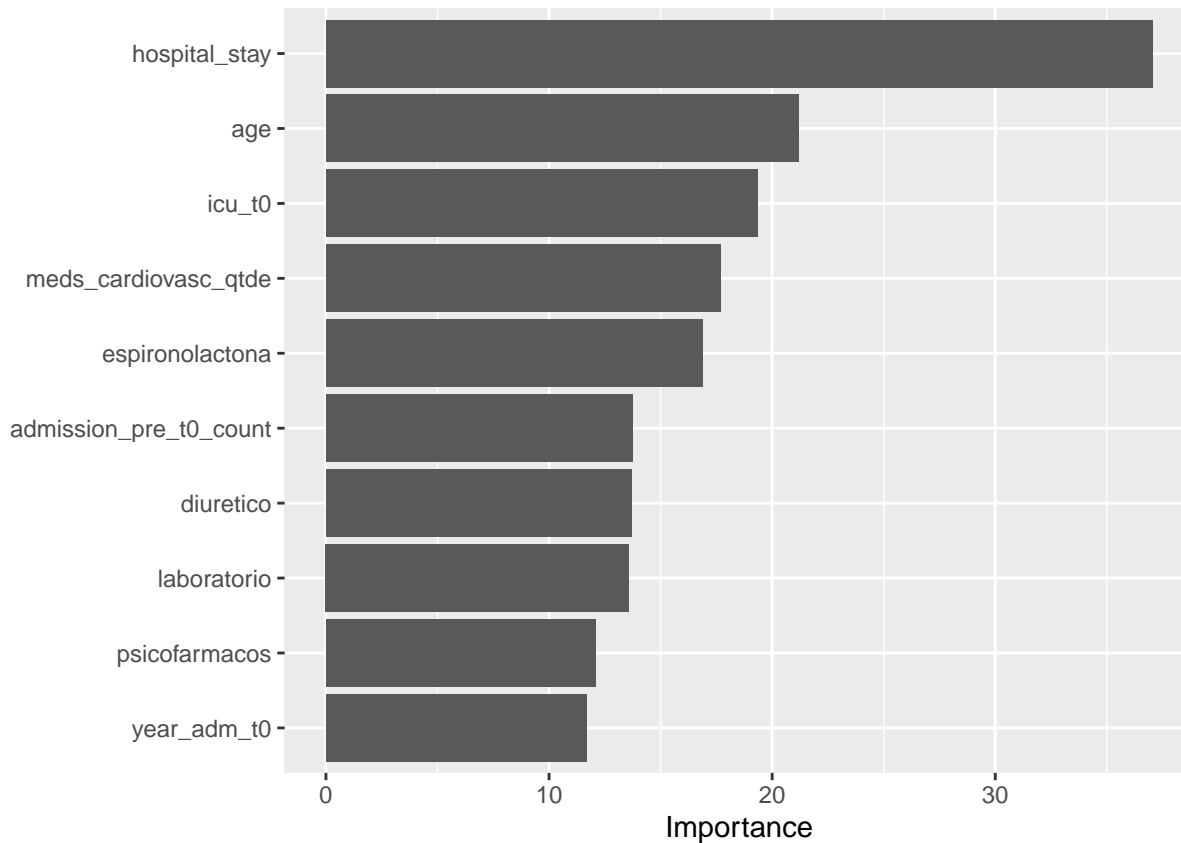
```

## [1] "Optimal Threshold: 0.03"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 3848  134
## 1  638  110
##
##              Accuracy : 0.8368
##              95% CI   : (0.8259, 0.8472)
##    No Information Rate : 0.9484
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1561
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.8578

```

```
##          Specificity : 0.4508
##          Pos Pred Value : 0.9663
##          Neg Pred Value : 0.1471
##          Prevalence : 0.9484
##          Detection Rate : 0.8135
##          Detection Prevalence : 0.8419
##          Balanced Accuracy : 0.6543
##
##          'Positive' Class : 0
##
```

```
extract_vip(final_tree_fit, pred_wrapper = predict,
            reference_class = "0", use_matrix = FALSE,
            method = 'model')
```



```
# extract_vip(final_tree_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 4.259

Random Forest

```
rf_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
        data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_impute_mean(all_numeric_predictors())

rf_spec <-
```



```

rand_forest(mtry = tune(),
            trees = tune(),
            min_n = tune()) %>%
set_mode("classification") %>%
set_engine("randomForest",
            probability = TRUE,
            nthread = 8)

rf_grid <- grid_latin_hypercube(mtry(range = c(1L, 50L)),
                               trees(range = c(100L, 300L)),
                               min_n(),
                               size = grid_size)

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

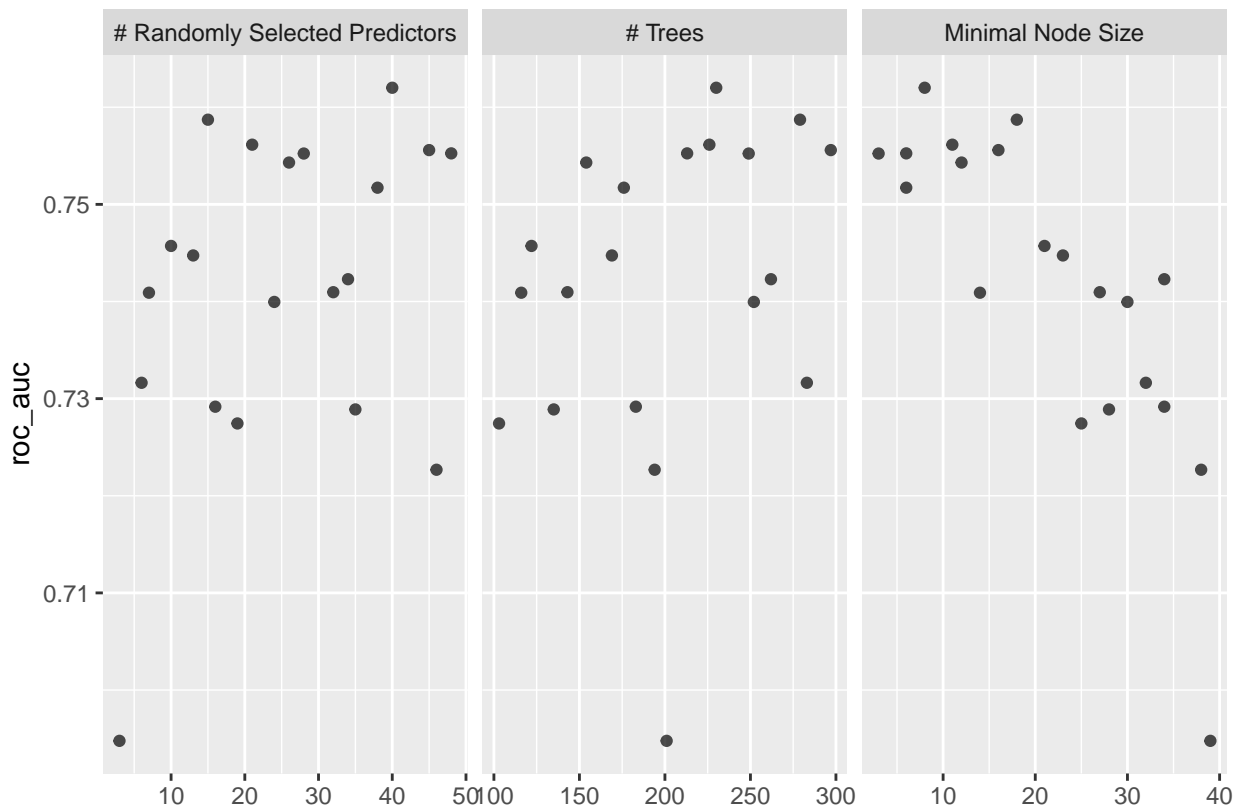
rf_tune <-
  rf_workflow %>%
  tune_grid(resamples = df_folds,
            grid = rf_grid)

rf_tune %>%
  collect_metrics()

## # A tibble: 40 x 9
##   mtry trees min_n .metric .estimator mean n std_err .config
##   <int> <int> <int> <chr>   <chr>   <dbl> <int>   <dbl> <chr>
## 1     48   213     6 accuracy binary    0.953    10 0.00136 Preprocessor1_Model01
## 2     48   213     6 roc_auc  binary    0.755    10 0.00906 Preprocessor1_Model01
## 3      6   283    32 accuracy binary    0.954    10 0.00132 Preprocessor1_Model02
## 4      6   283    32 roc_auc  binary    0.732    10 0.0115  Preprocessor1_Model02
## 5     38   176     6 accuracy binary    0.954    10 0.00135 Preprocessor1_Model03
## 6     38   176     6 roc_auc  binary    0.752    10 0.0105  Preprocessor1_Model03
## 7     34   262    34 accuracy binary    0.954    10 0.00136 Preprocessor1_Model04
## 8     34   262    34 roc_auc  binary    0.742    10 0.0104  Preprocessor1_Model04
## 9     46   194    38 accuracy binary    0.954    10 0.00131 Preprocessor1_Model05
## 10    46   194    38 roc_auc  binary    0.723    10 0.0115  Preprocessor1_Model05
## # ... with 30 more rows
## # i Use 'print(n = ...)' to see more rows

autoplot(rf_tune, metric = "roc_auc")

```



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

```
## # A tibble: 5 x 9
##   mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>    <dbl> <int>   <dbl> <chr>
## 1    40   230     8 roc_auc binary  0.762    10 0.00929 Preprocessor1_Model108
## 2    15   279    18 roc_auc binary  0.759    10 0.00948 Preprocessor1_Model117
## 3    21   226    11 roc_auc binary  0.756    10 0.0115  Preprocessor1_Model111
## 4    45   297    16 roc_auc binary  0.756    10 0.00945 Preprocessor1_Model119
## 5    48   213     6 roc_auc binary  0.755    10 0.00906 Preprocessor1_Model101
```

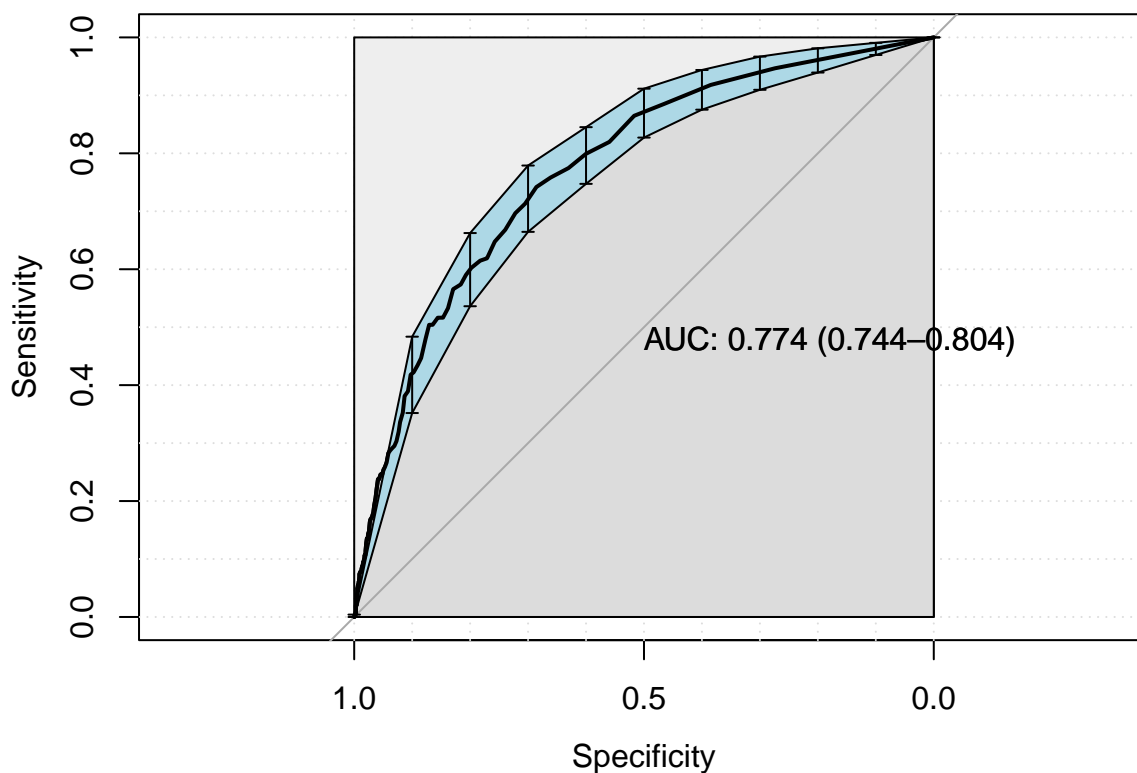
```
best_rf <- rf_tune %>%
  select_best("roc_auc")
```

```
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)
```

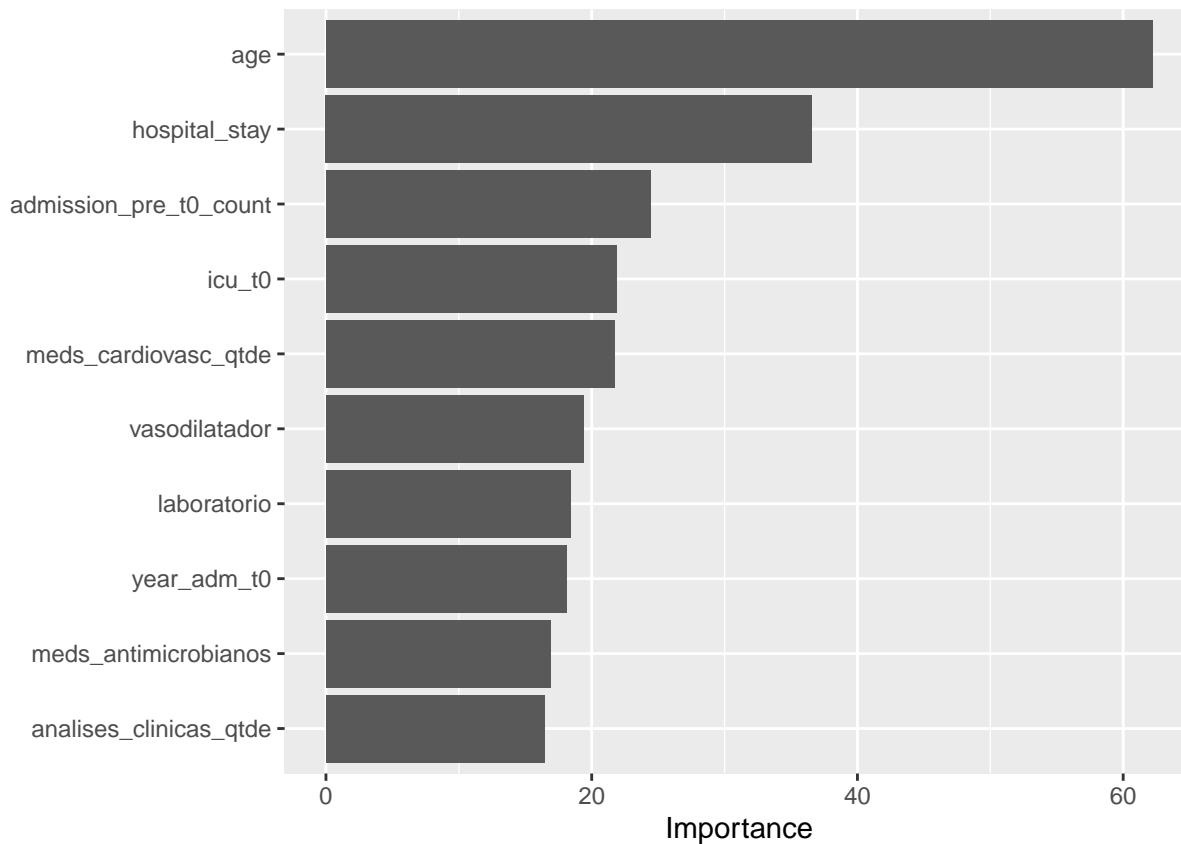
```
rf_auc <- validation(final_rf_fit, df_test)
```



```
## [1] "Optimal Threshold: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 3076   63
##    1 1410  181
##
##              Accuracy : 0.6886
##              95% CI   : (0.6752, 0.7018)
##    No Information Rate : 0.9484
##    P-Value [Acc > NIR] : 1
##
##              Kappa : 0.1184
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6857
##              Specificity : 0.7418
##    Pos Pred Value : 0.9799
##    Neg Pred Value : 0.1138
##    Prevalence : 0.9484
##    Detection Rate : 0.6503
##    Detection Prevalence : 0.6636
##    Balanced Accuracy : 0.7137
##
##    'Positive' Class : 0
##
```

```
pfun_rf <- function(object, newdata) predict(object, data = newdata)

extract_vip(final_rf_fit, pred_wrapper = predict,
             reference_class = "1", use_matrix = FALSE,
             method = 'model')
```



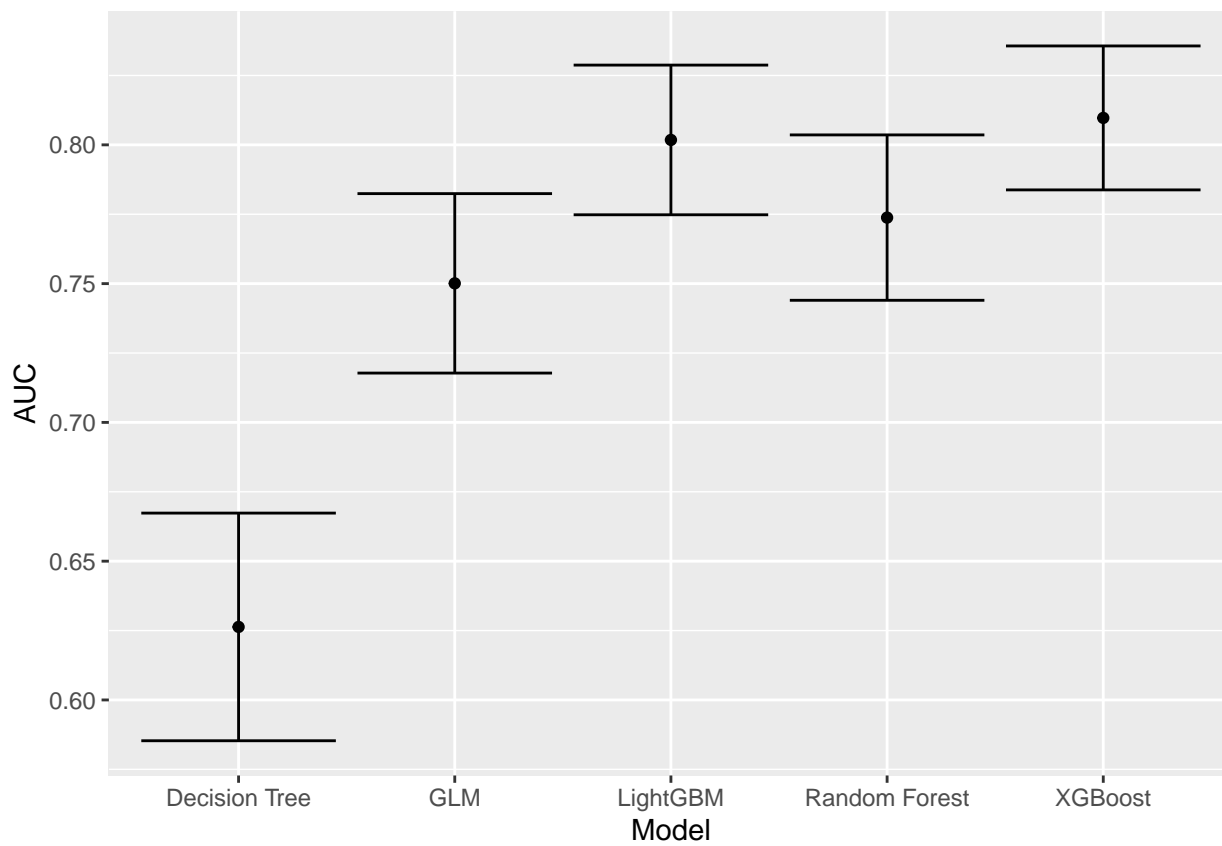
```
# extract_vip(final_rf_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 64.721

Models Comparison

```
if (RUN_ALL_MODELS) {
  df_auc <- tibble::tribble(
    ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
    'XGBoost', as.numeric(xgboost_auc$auc), xgboost_auc$ci[1], xgboost_auc$ci[3],
    'LightGBM', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3],
    'GLM', as.numeric(glmnet_auc$auc), glmnet_auc$ci[1], glmnet_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$ci[3]
  ) %>%
    mutate(Target = outcome_column)
} else {
  df_auc <- tibble::tribble(
    ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
    'LightGBM', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3]
  ) %>%
    mutate(Target = outcome_column)
}

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



```
write_csv(df_auc, sprintf("./auxiliar/model_selection/performance/%s.csv", outcome_column))
```

Minutes to run: 0.002