Model Selection - readmission_30d

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= readmission_30d
## .  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: 0.029

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. education_level
## 02. underlying_heart_disease
## 03. heart_disease
## 04. nyha_basal
## 05. prior_mi
## 06. heart_failure
## 07. transplant
## 08. endocardites
## 09. hemodialysis
## 10. comorbidities_count
## 11. procedure_type_1
## 12. reop_type_1
## 13. procedure_type_new
```

14. cied_final_1
15. cied_final_group_1
16. admission_pre_t0_count
17. admission_pre_t0_180d
18. icu_t0
19. dialysis_t0
20. admission_t0_emergency
21. aco
22. antiarritmico
23. betabloqueador
24. ieca_bra
25. dva
26. digoxina
27. estatina
28. diuretico
29. vasodilatador
30. insuf_cardiaca
31. espironolactona
32. bloq_calcio
33. antiplaquetario_ev
34. insulina
35. anticonvulsivante
36. psicofarmacos
37. antifungico
38. antiviral
39. classe_meds_qtde
40. meds_cardiovasc_qtde
41. meds_antimicrobianos
42. ventilacao_mecanica
43. cec
44. transplante_cardiaco
45. outros_proced_cirurgicos
46. icp
47. intervencao_cv
48. cateterismo
49. eletrofisiologia
50. cateter_venoso_central
51. proced_invasivos_qtde
52. cve_desf
53. transfusao
54. equipe_multiprof
55. holter
56. metodos_graficos_qtde
57. laboratorio
58. cultura
59. analises_clinicas_qtde
60. citologia
61. biopsia
62. histopatologia_qtde
63. angio_rm
64. angio_tc
65. cintilografia
66. ecocardiograma
67. endoscopia
68. flebografia
69. pet_ct
70. ultrassom
71. tomografia
72. ressonancia
73. exames_imagem_qtde
74. bic

```
## 75. mpp
## 76. 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: 0.004

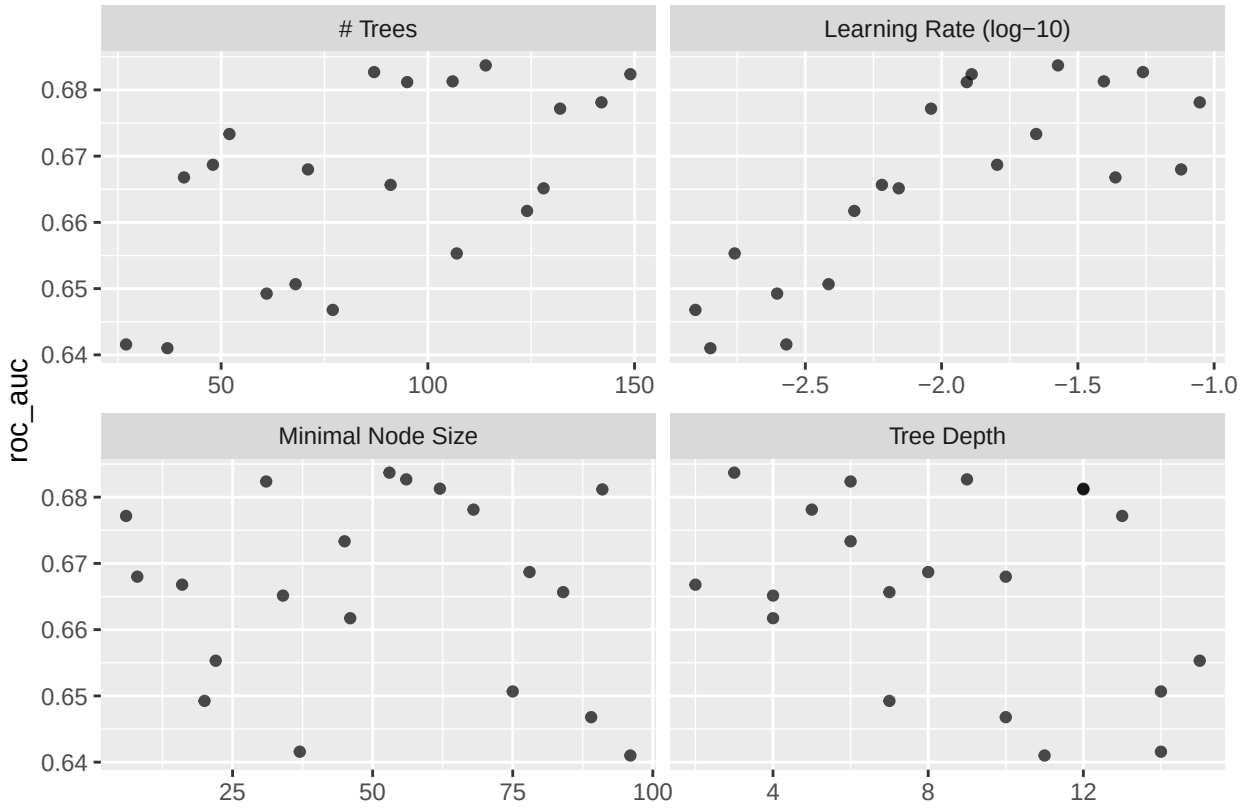
Boosted Tree (XGBoost)

```
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)
```

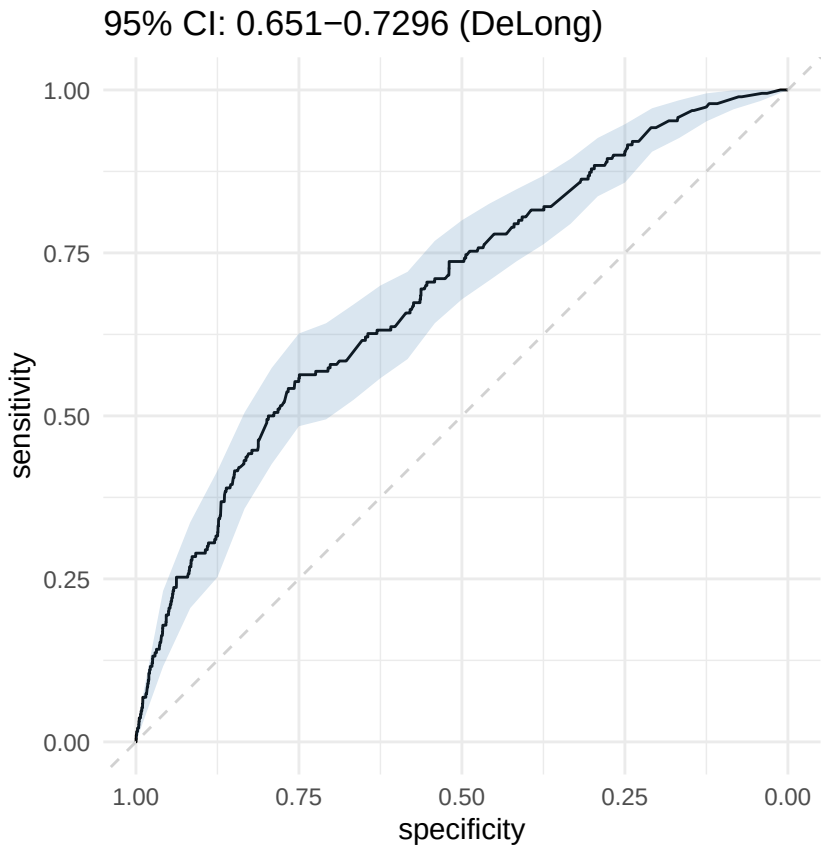
```
xgboost_tune %>%
  show_best("roc_auc")
```

```
## # A tibble: 5 x 10
##   trees min_n tree_depth learn_rate .metric .estimator  mean
##   <int> <int>      <int>      <dbl> <chr>   <chr>      <dbl>
## 1   114    53          3     0.0267 roc_auc binary     0.684
## 2    87    56          9     0.0548 roc_auc binary     0.683
## 3   149    31          6     0.0129 roc_auc binary     0.682
## 4   106    62         12     0.0394 roc_auc binary     0.681
## 5    95    91         12     0.0124 roc_auc binary     0.681
## # i 3 more variables: n <int>, std_err <dbl>, .config <chr>
```

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

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

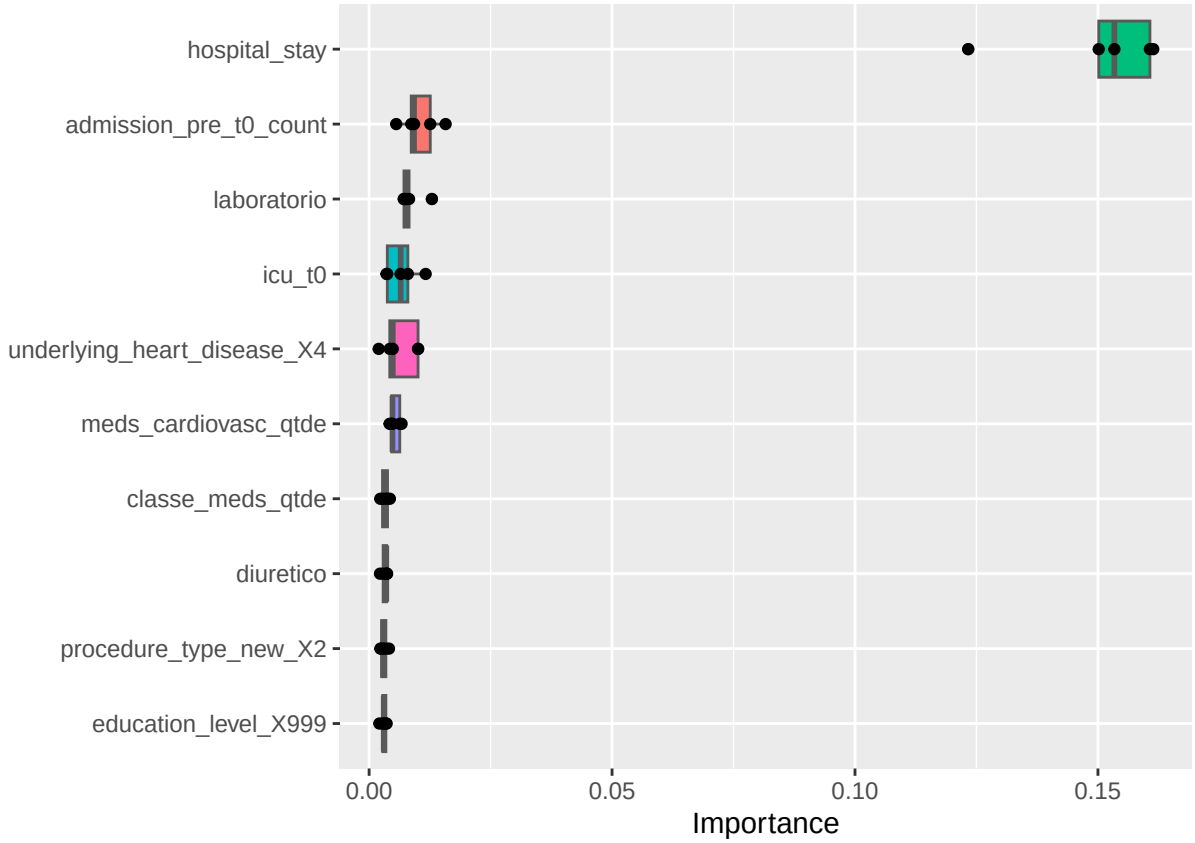



```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

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: 0.07"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3402   83
##    1 1138  107
##
##                Accuracy : 0.7419
##                  95% CI : (0.7291, 0.7543)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0854
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.74934
##             Specificity : 0.56316
##          Pos Pred Value : 0.97618
##          Neg Pred Value : 0.08594
##              Prevalence : 0.95983
##          Detection Rate : 0.71924
##    Detection Prevalence : 0.73679
##       Balanced Accuracy : 0.65625
##
##        '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: 3.83

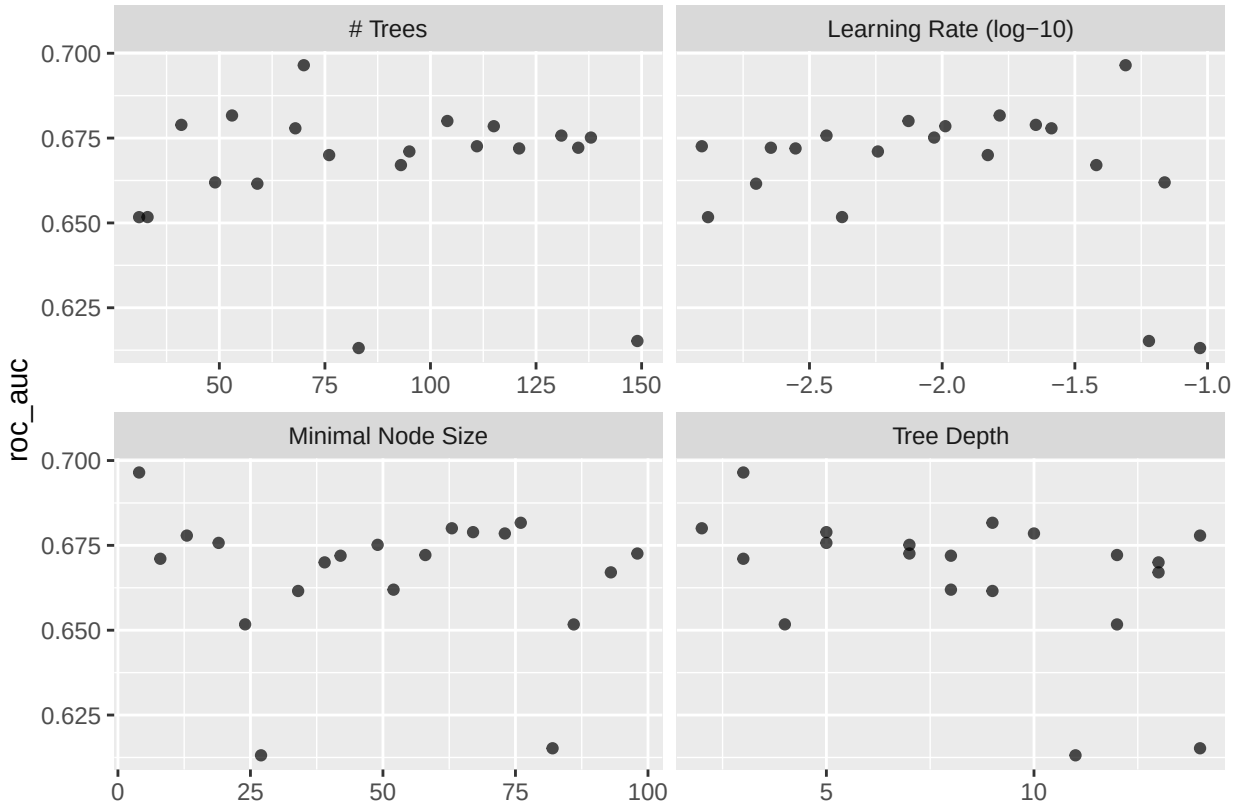
Boosted Tree (LightGBM)

```
lightgbm_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())

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")
```

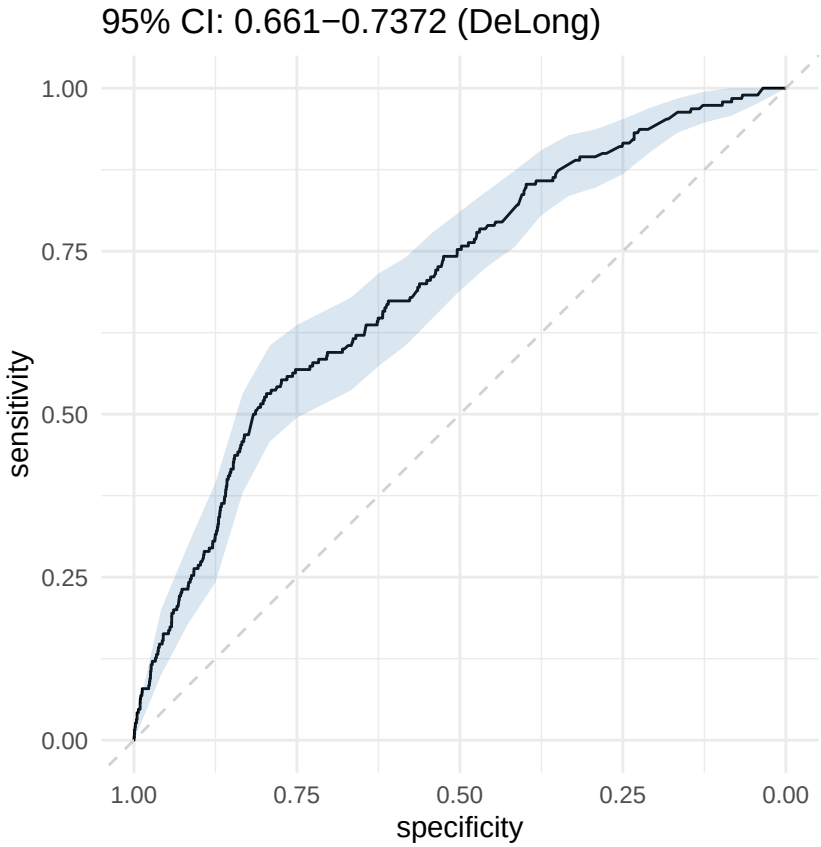
```
## # A tibble: 5 x 10
##   trees min_n tree_depth learn_rate .metric .estimator  mean
##   <int> <int>      <int>      <dbl> <chr>   <chr>      <dbl>
## 1    70     4          3    0.0491  roc_auc binary     0.696
## 2    53    76          9    0.0165  roc_auc binary     0.682
## 3   104    63          2    0.00747 roc_auc binary     0.680
## 4    41    67          5    0.0225  roc_auc binary     0.679
## 5   115    73         10    0.0103  roc_auc binary     0.678
## # i 3 more variables: n <int>, std_err <dbl>, .config <chr>
```

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

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




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

```r
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 3617   89
##    1  923  101
##
##                Accuracy : 0.786
##                  95% CI : (0.7741, 0.7977)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1058
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.79670
##             Specificity : 0.53158
##          Pos Pred Value : 0.97598
##          Neg Pred Value : 0.09863
##              Prevalence : 0.95983
##          Detection Rate : 0.76469
##    Detection Prevalence : 0.78351
##       Balanced Accuracy : 0.66414
```

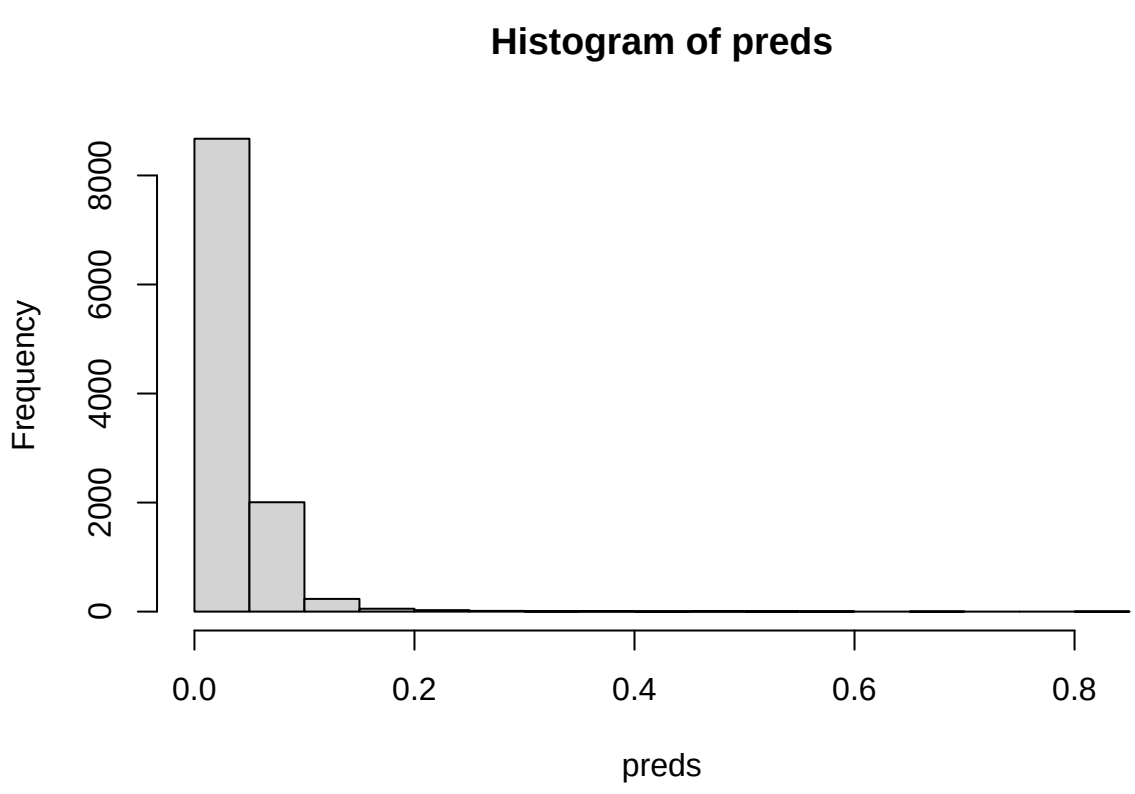
```
##
##        'Positive' Class : 0
##
```

```
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= 70
## .  min_n = integer 1= 4
## .  tree_depth = integer 1= 3
## .  learn_rate = double 1= 0.049118
```

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

Minutes to run: 3.053

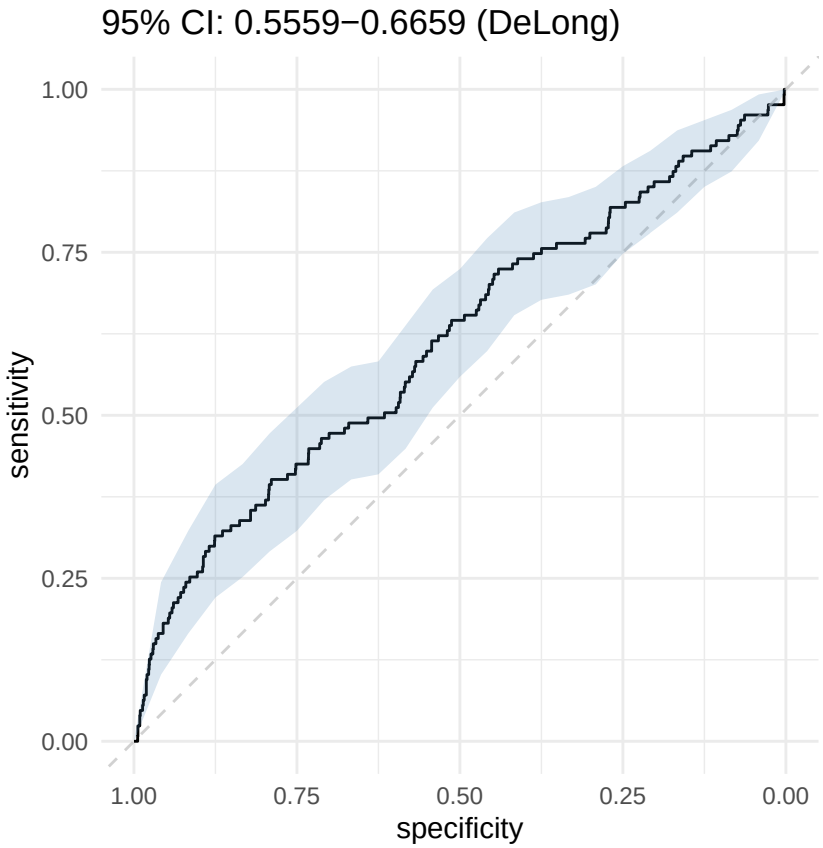


Minutes to run: 0.003

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())
```

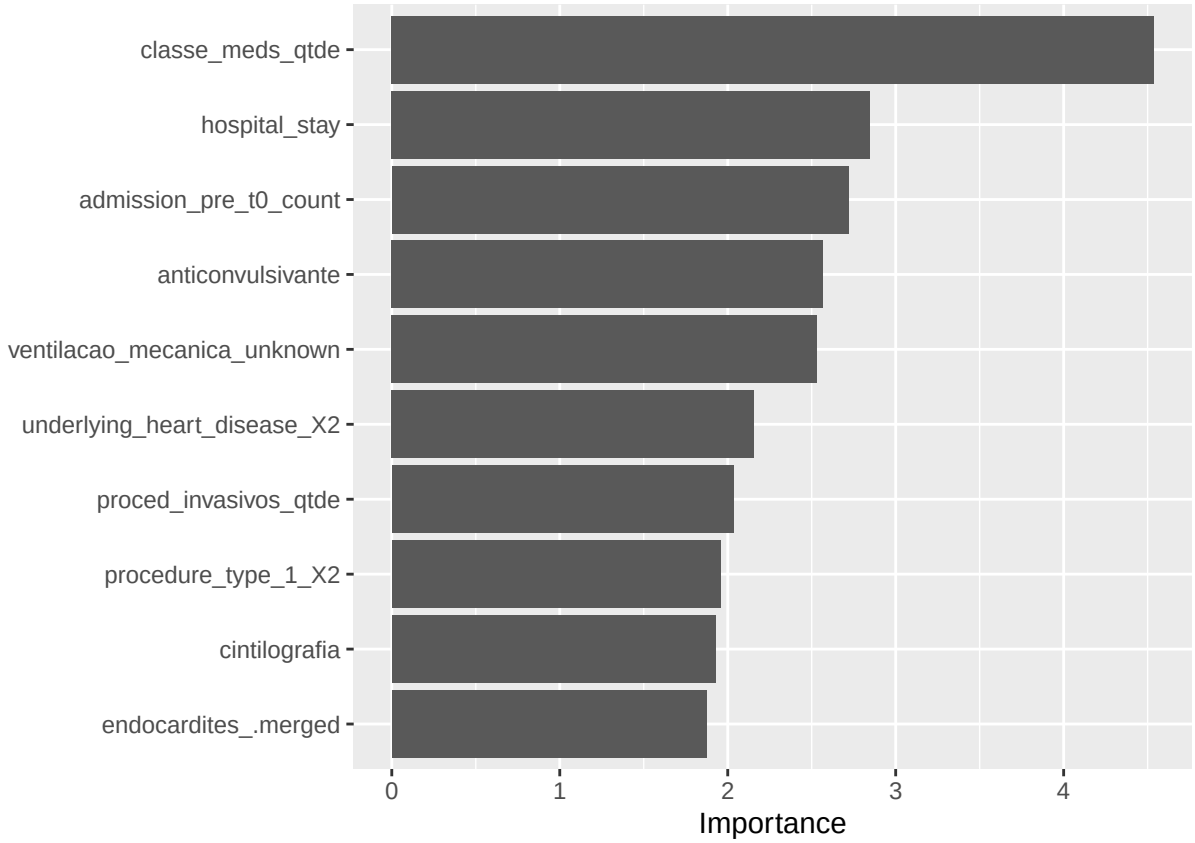
```r
glmnet_spec <-
  logistic_reg(penalty = 0) %>%
  set_mode("classification") %>%
  set_engine("glm")

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: 0.06"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 2444   87
##    1  346   40
##
##                Accuracy : 0.8516
##                  95% CI : (0.8381, 0.8643)
##     No Information Rate : 0.9565
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0968
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8760
```

```
##             Specificity : 0.3150
##          Pos Pred Value : 0.9656
##          Neg Pred Value : 0.1036
##              Prevalence : 0.9565
##          Detection Rate : 0.8378
##    Detection Prevalence : 0.8677
##       Balanced Accuracy : 0.5955
##
##        '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')
```

Importance

Minutes to run: 0.205

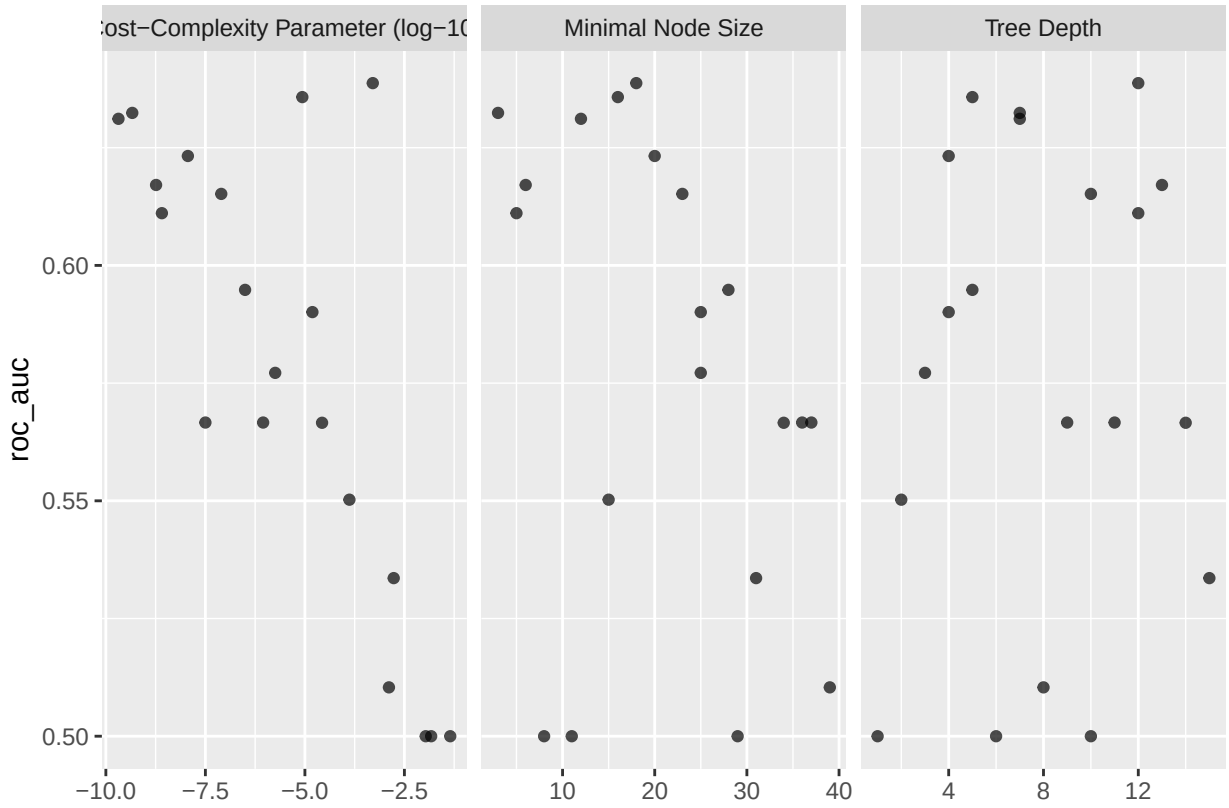
Decision Tree

```
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
##              <dbl>      <int> <int> <chr>   <chr>      <dbl>
##  1        4.61e-10          7     3 accura~ binary     0.961
##  2        4.61e-10          7     3 roc_auc binary     0.632
##  3        1.54e- 5          4    25 accura~ binary     0.962
##  4        1.54e- 5          4    25 roc_auc binary     0.590
##  5        2.07e-10          7    12 accura~ binary     0.960
##  6        2.07e-10          7    12 roc_auc binary     0.631
##  7        1.70e- 3         15    31 accura~ binary     0.962
##  8        1.70e- 3         15    31 roc_auc binary     0.534
##  9        4.48e- 2         10    29 accura~ binary     0.963
## 10        4.48e- 2         10    29 roc_auc binary     0.5
## # i 30 more rows
## # i 3 more variables: n <int>, std_err <dbl>, .config <chr>
```

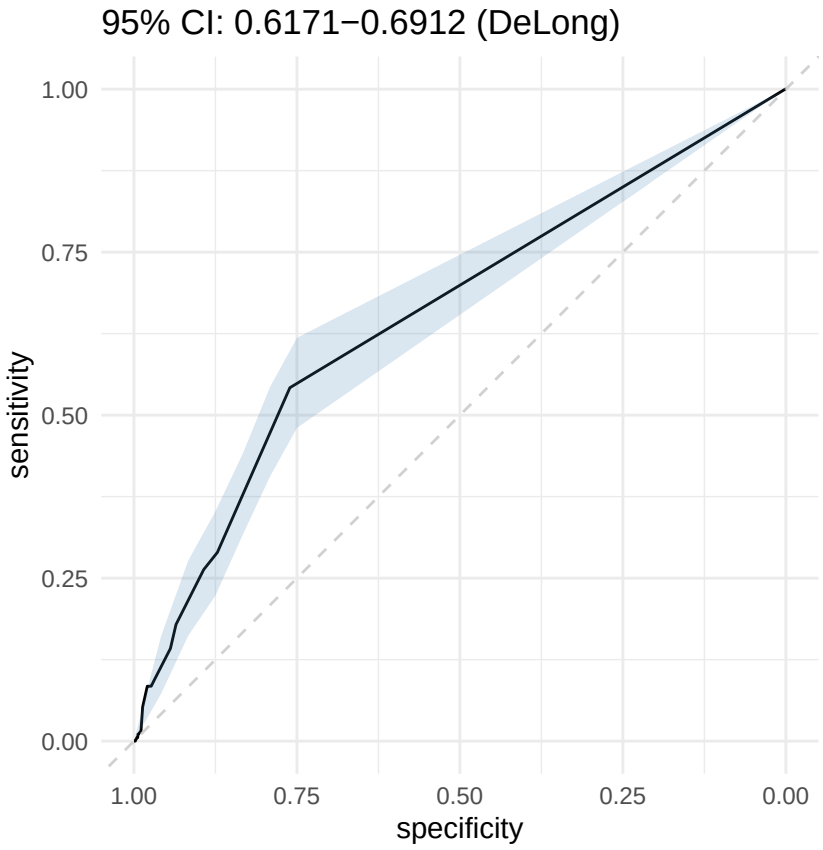
```
autoplot(tree_tune, metric = "roc_auc")
```




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

```
## # A tibble: 5 x 9
##   cost_complexity tree_depth min_n .metric .estimator  mean
##             <dbl>      <int> <int> <chr>   <chr>      <dbl>
## 1        5.07e- 4         12    18 roc_auc binary     0.639
## 2        8.62e- 6          5    16 roc_auc binary     0.636
## 3        4.61e-10          7     3 roc_auc binary     0.632
## 4        2.07e-10          7    12 roc_auc binary     0.631
## 5        1.15e- 8          4    20 roc_auc binary     0.623
## # i 3 more variables: n <int>, std_err <dbl>, .config <chr>
```

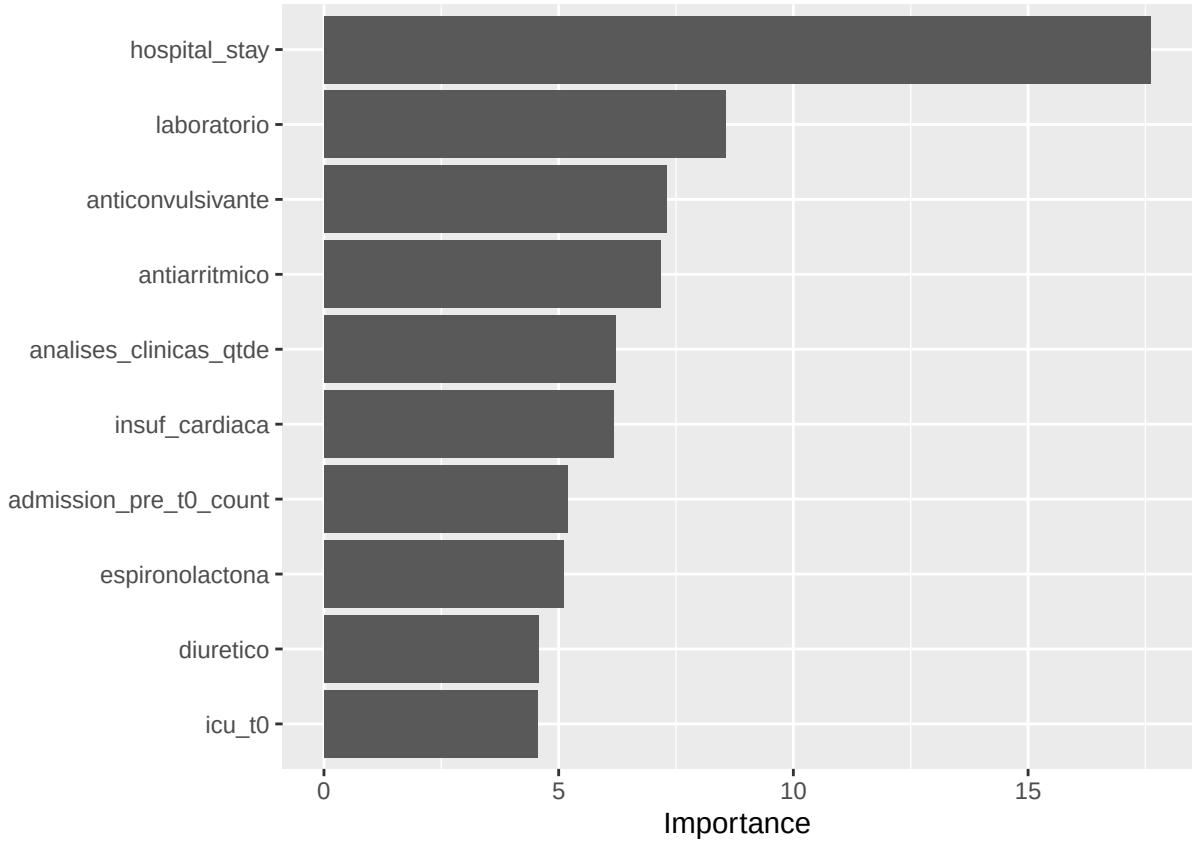
```r
best_tree <- tree_tune %>%
  select_best("roc_auc")

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: 0.04"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3454   87
##    1 1086  103
##
##                Accuracy : 0.752
##                  95% CI : (0.7394, 0.7643)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0861
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.76079
```

```
##             Specificity : 0.54211
##          Pos Pred Value : 0.97543
##          Neg Pred Value : 0.08663
##              Prevalence : 0.95983
##          Detection Rate : 0.73023
##    Detection Prevalence : 0.74863
##       Balanced Accuracy : 0.65145
##
##        'Positive' Class : 0
##
```

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




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

Minutes to run: 3.091

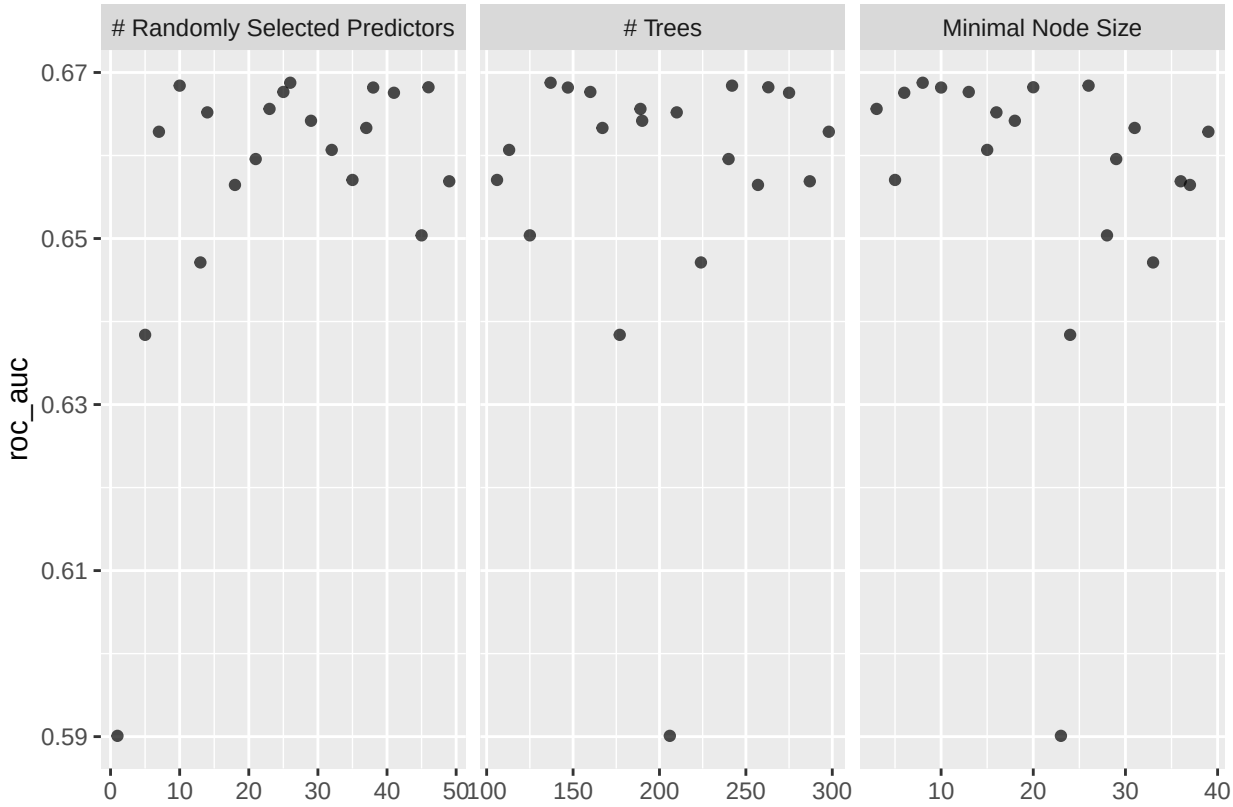
Random Forest

```r
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
##    <int> <int> <int> <chr>    <chr>      <dbl> <int>   <dbl>
##  1    23   189     3 accuracy binary     0.963    10 0.00215
##  2    23   189     3 roc_auc  binary     0.666    10 0.0154
##  3    21   240    29 accuracy binary     0.962    10 0.00221
##  4    21   240    29 roc_auc  binary     0.660    10 0.0113
##  5    46   263    20 accuracy binary     0.963    10 0.00220
##  6    46   263    20 roc_auc  binary     0.668    10 0.0104
##  7    41   275     6 accuracy binary     0.962    10 0.00222
##  8    41   275     6 roc_auc  binary     0.668    10 0.0136
##  9    38   147    10 accuracy binary     0.962    10 0.00206
## 10    38   147    10 roc_auc  binary     0.668    10 0.0135
## # i 30 more rows
## # i 1 more variable: .config <chr>
```

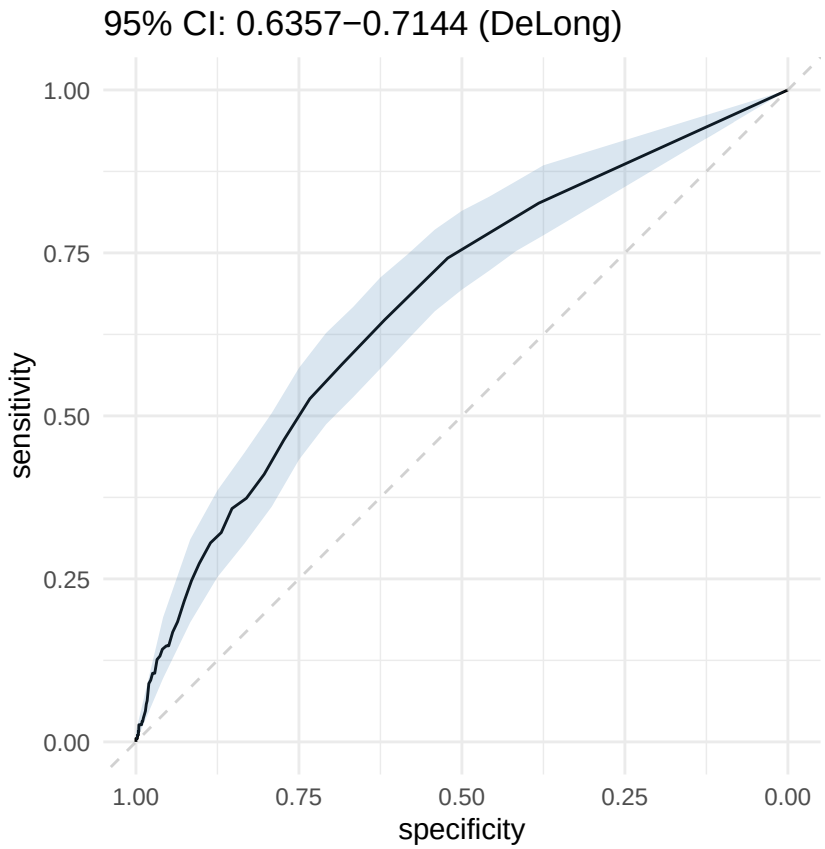
```
autoplot(rf_tune, metric = "roc_auc")
```




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

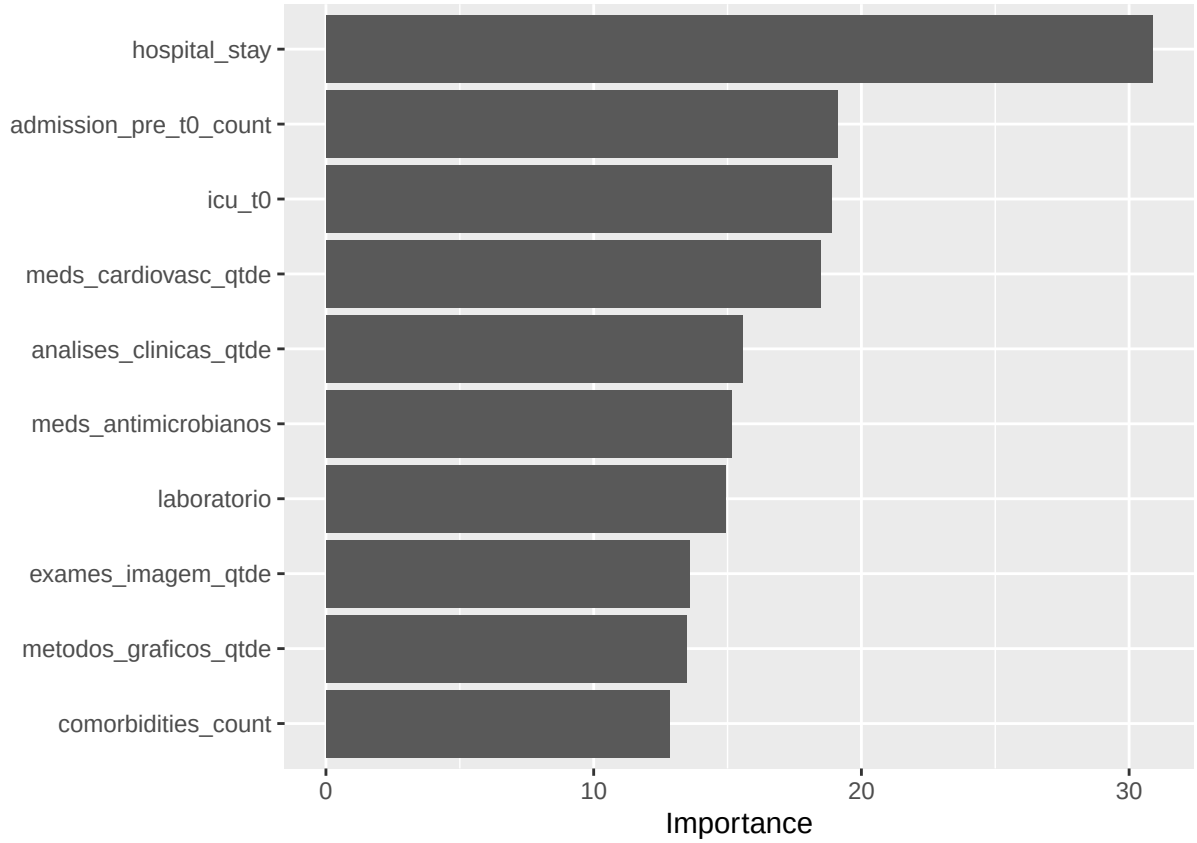
```
## # A tibble: 5 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl>
## 1    26   137     8 roc_auc binary     0.669    10  0.0147
## 2    10   242    26 roc_auc binary     0.668    10  0.0108
## 3    46   263    20 roc_auc binary     0.668    10  0.0104
## 4    38   147    10 roc_auc binary     0.668    10  0.0135
## 5    25   160    13 roc_auc binary     0.668    10  0.0122
## # i 1 more variable: .config <chr>
```

```
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: 0.02"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 2808   67
##    1 1732  123
##
##                Accuracy : 0.6197
##                  95% CI : (0.6057, 0.6335)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0511
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.61850
##             Specificity : 0.64737
##          Pos Pred Value : 0.97670
##          Neg Pred Value : 0.06631
##              Prevalence : 0.95983
##          Detection Rate : 0.59366
##    Detection Prevalence : 0.60782
##       Balanced Accuracy : 0.63294
##
##        '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')
```

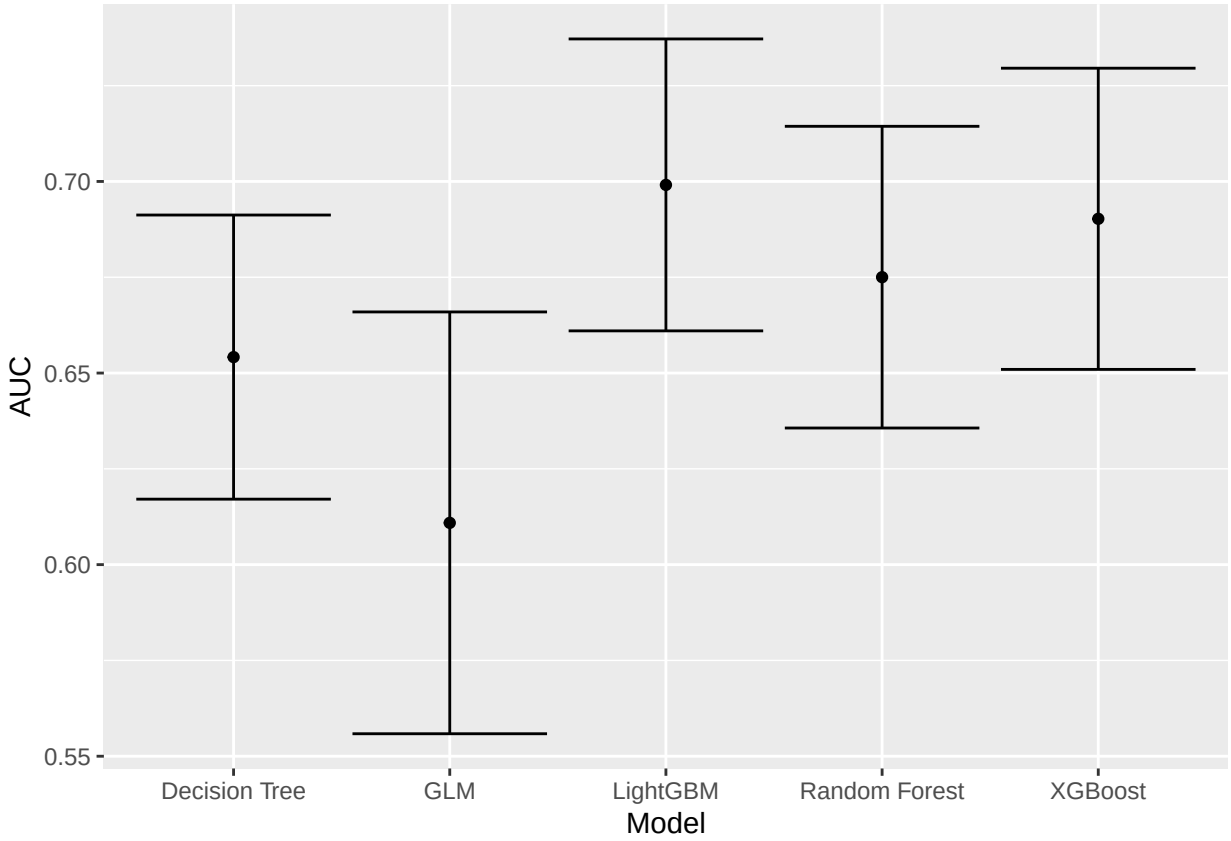



```
# extract_vip(final_rf_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 68.604

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: 0.008