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: 0.026

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: 0.005

```
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. espironolactona
## 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

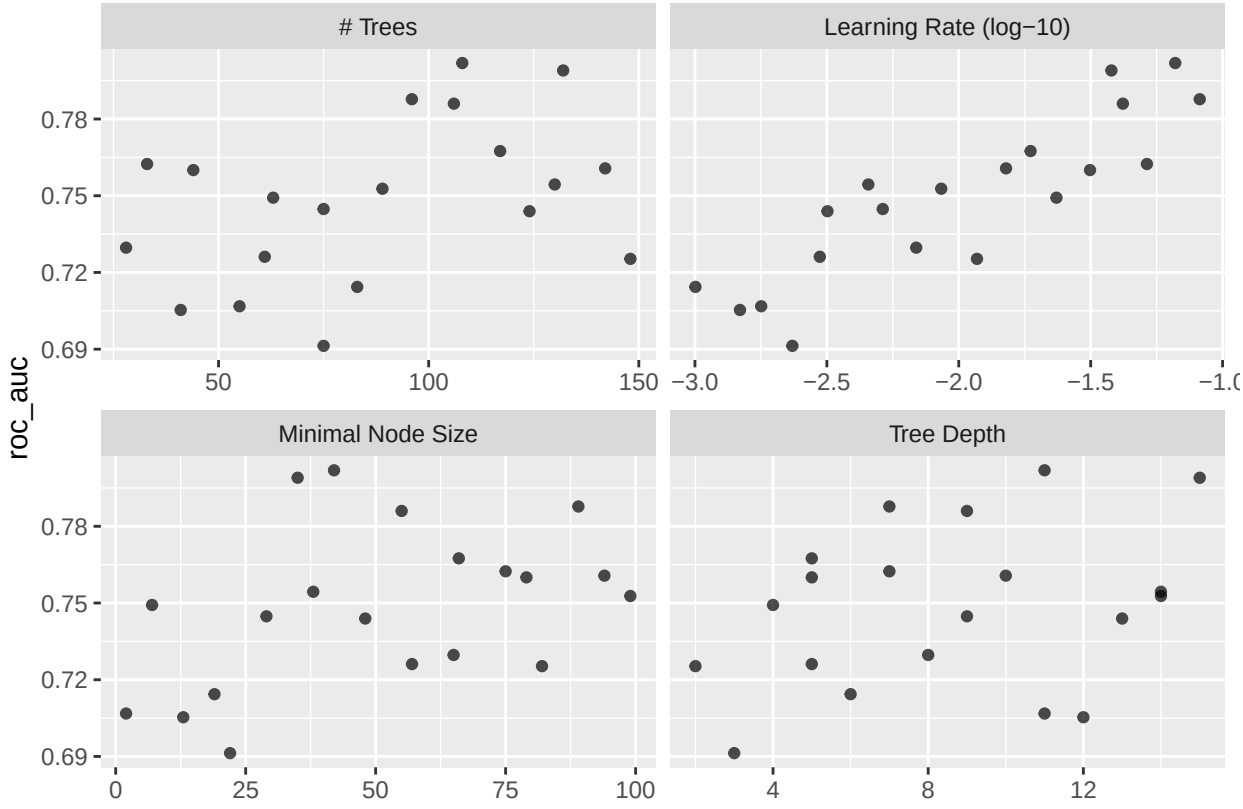
Boosted Tree (XGBoost)

```
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 <-
```

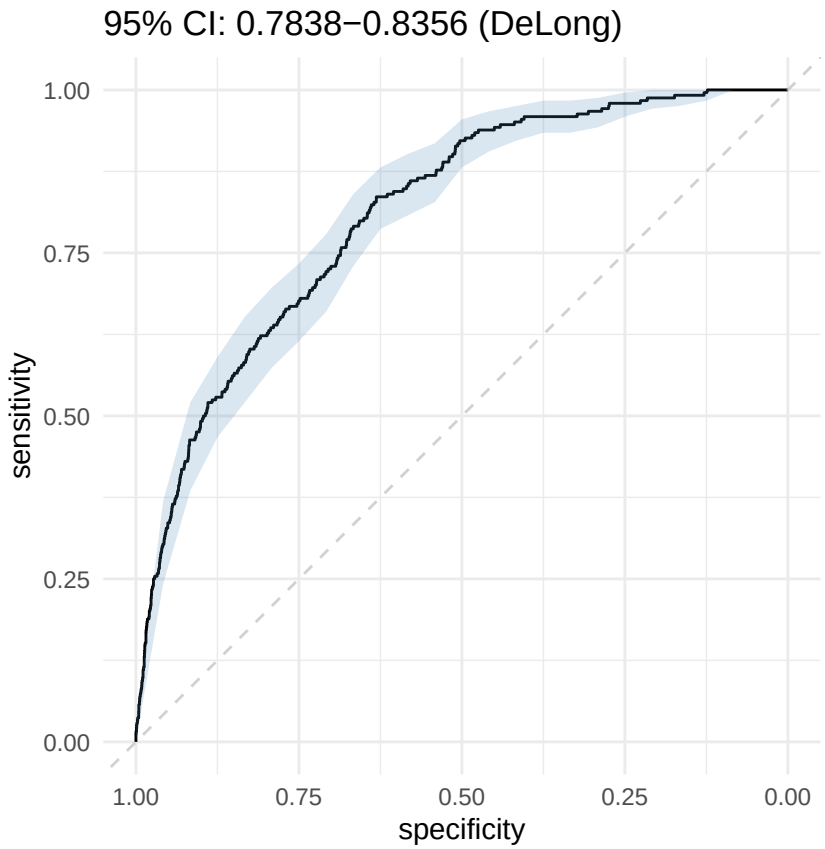
```
  xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)

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     0.802    10 0.00889 Preprocessor1_Model09
## 2   132    35         15     0.0379 roc_auc binary     0.799    10 0.00896 Preprocessor1_Model07
## 3    96    89          7     0.0819 roc_auc binary     0.788    10 0.00868 Preprocessor1_Model18
## 4   106    55          9     0.0419 roc_auc binary     0.786    10 0.00821 Preprocessor1_Model11
## 5   117    66          5     0.0187 roc_auc binary     0.767    10 0.00767 Preprocessor1_Model14
```

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

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

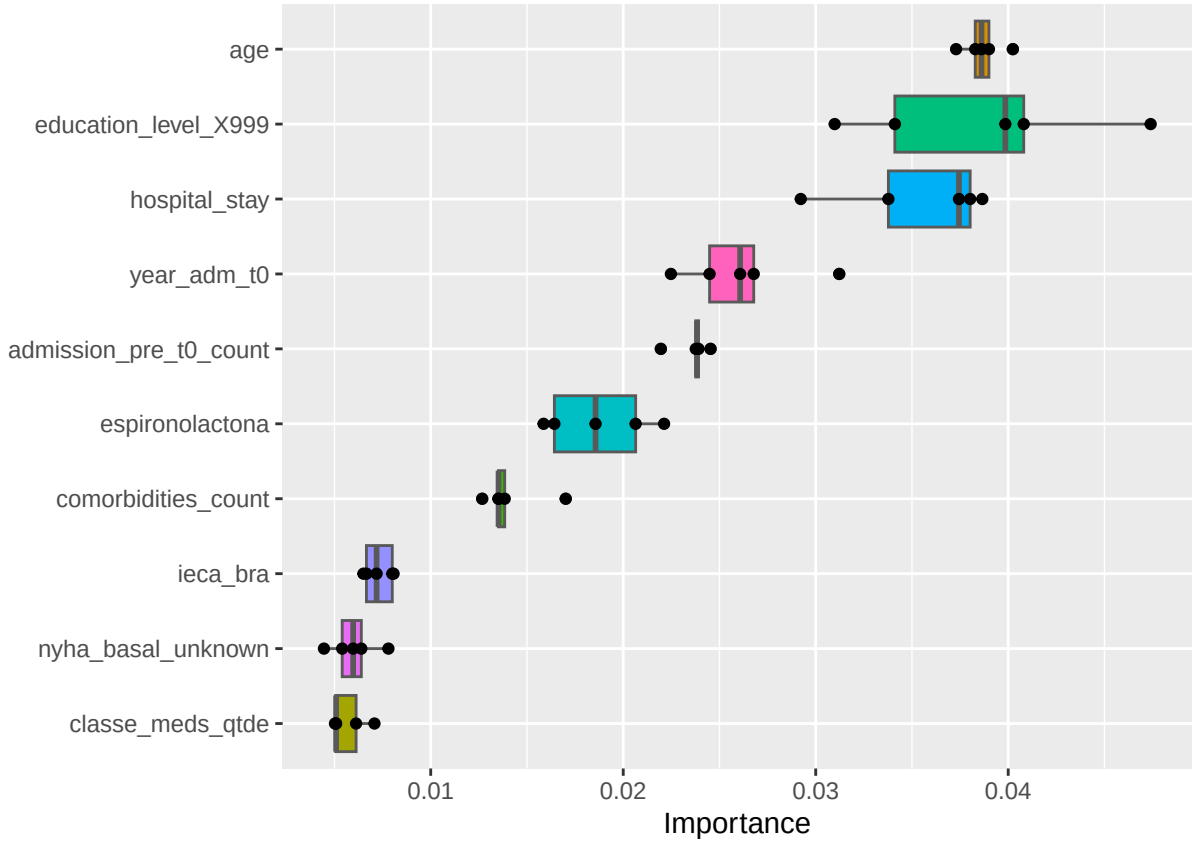



```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

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: 4.027

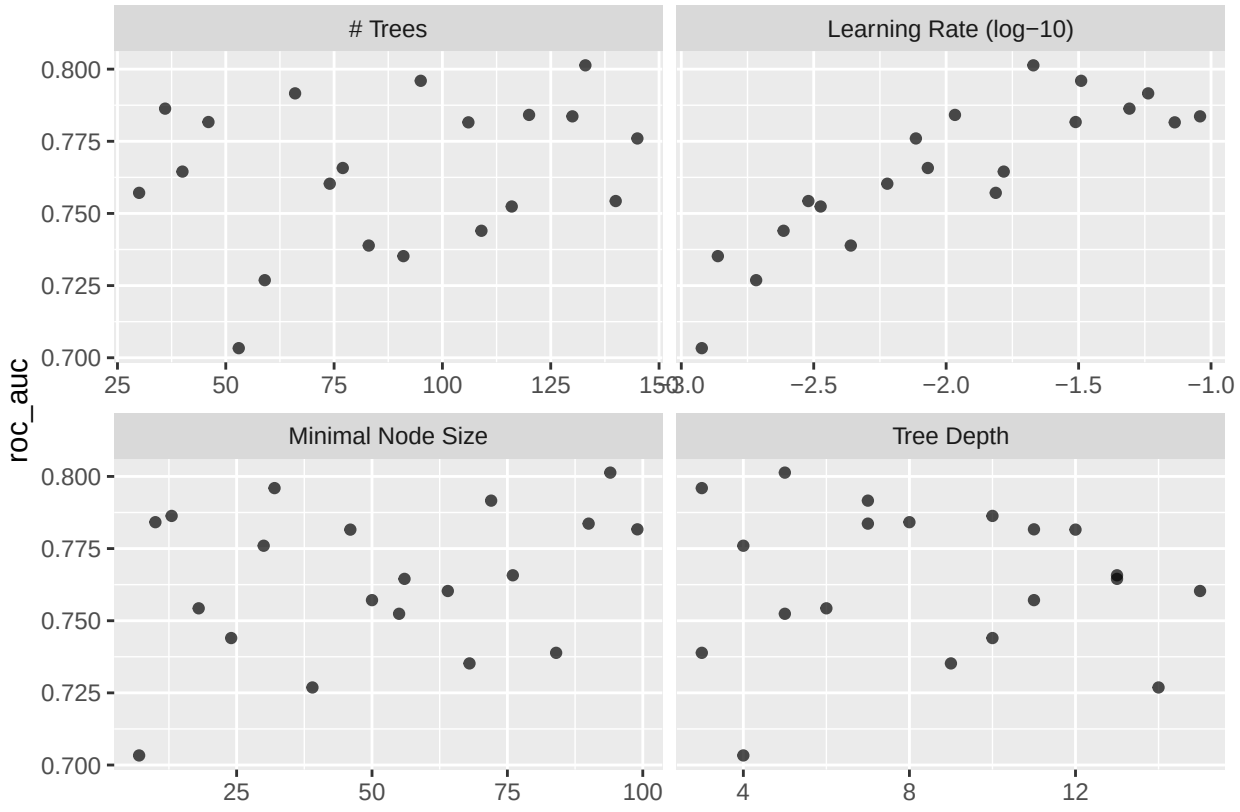
Boosted Tree (LightGBM)

```
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")
```

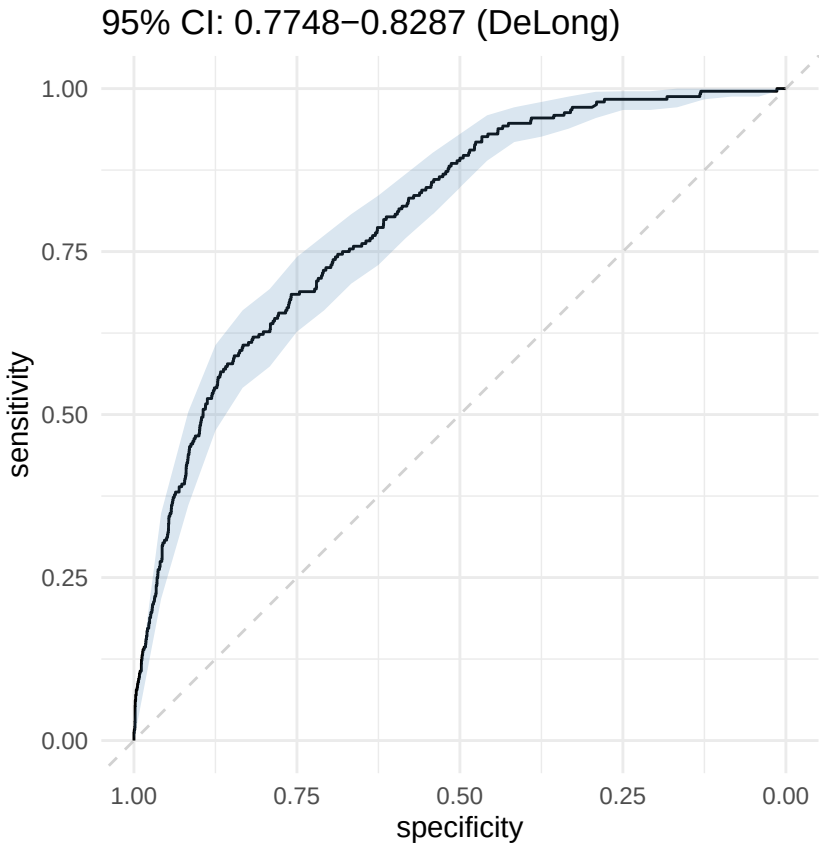
```
## # A tibble: 5 x 10
##   trees min_n tree_depth learn_rate .metric .estimator  mean     n std_err .config
##   <int> <int>      <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1   133    94          5     0.0213 roc_auc binary     0.801    10 0.00850 Preprocessor1_Model19
## 2    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
##
```

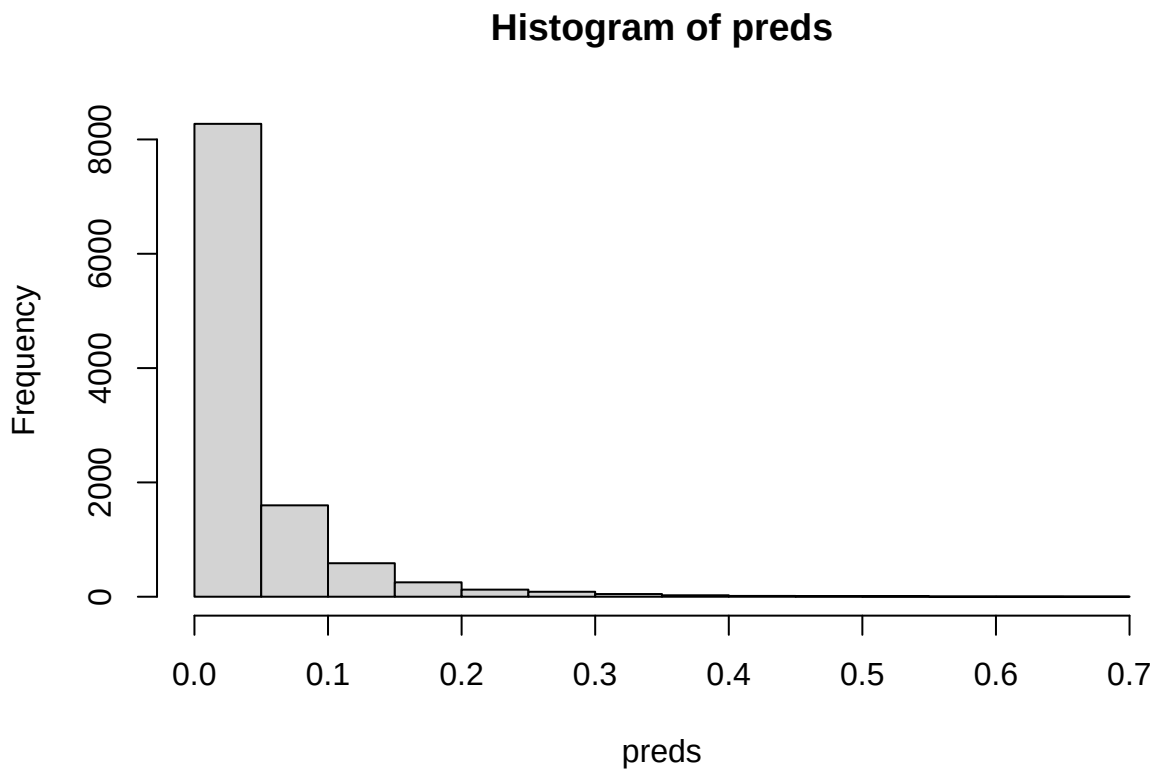
```
##        'Positive' Class : 0
##
```

```
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: 3.213

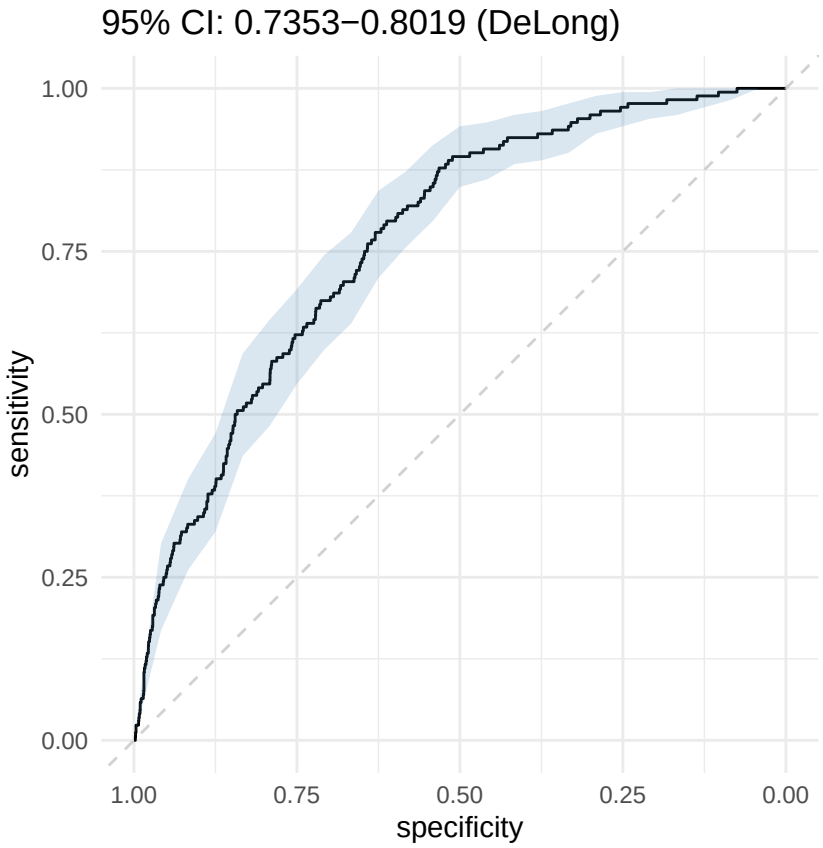


Minutes to run: 0.005

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())
```

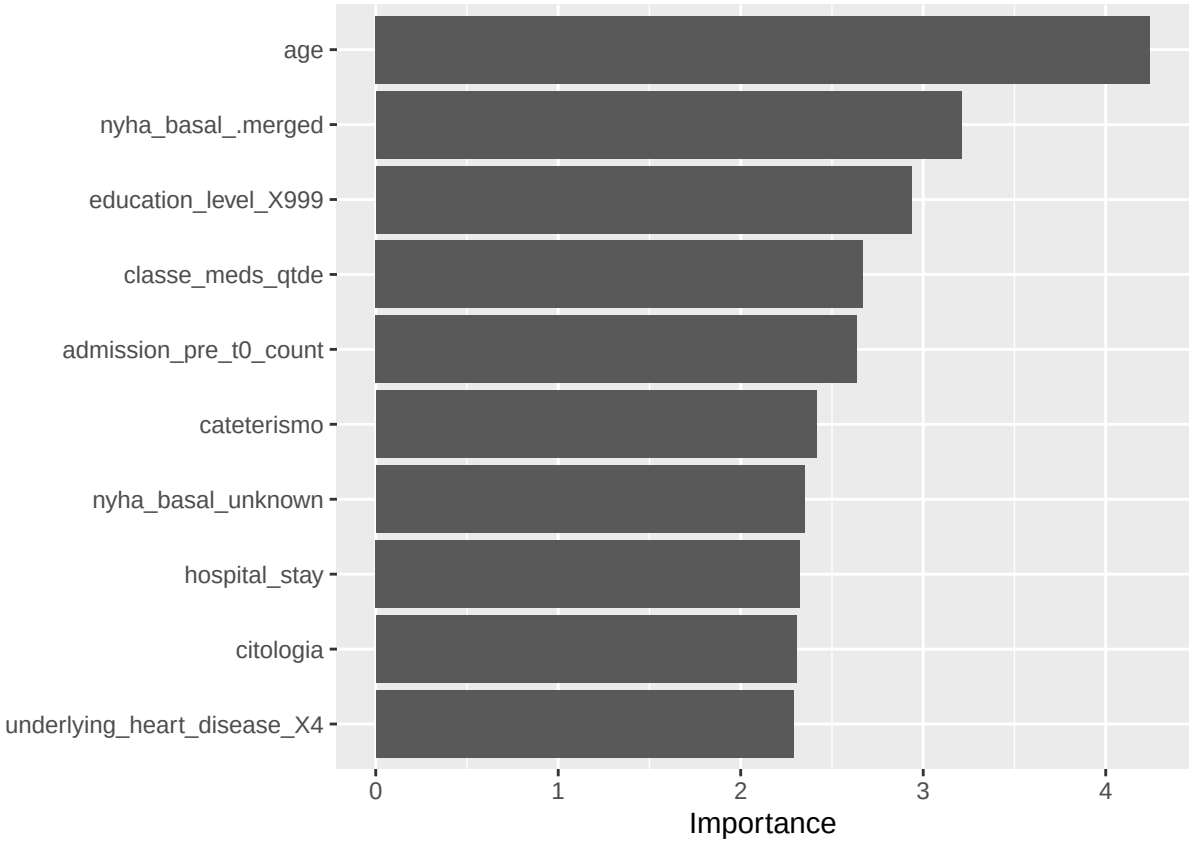
```r
glmnet_spec <-
  logistic_reg(penalty = 0) %>%
  set_mode("classification") %>%
  set_engine("glm")

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: 0.03"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 1491   21
##    1 1311  151
##
##                Accuracy : 0.5521
##                  95% CI : (0.534, 0.5701)
##     No Information Rate : 0.9422
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0907
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5321
```

```
##             Specificity : 0.8779
##          Pos Pred Value : 0.9861
##          Neg Pred Value : 0.1033
##              Prevalence : 0.9422
##          Detection Rate : 0.5013
##    Detection Prevalence : 0.5084
##       Balanced Accuracy : 0.7050
##
##        '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: 0.222

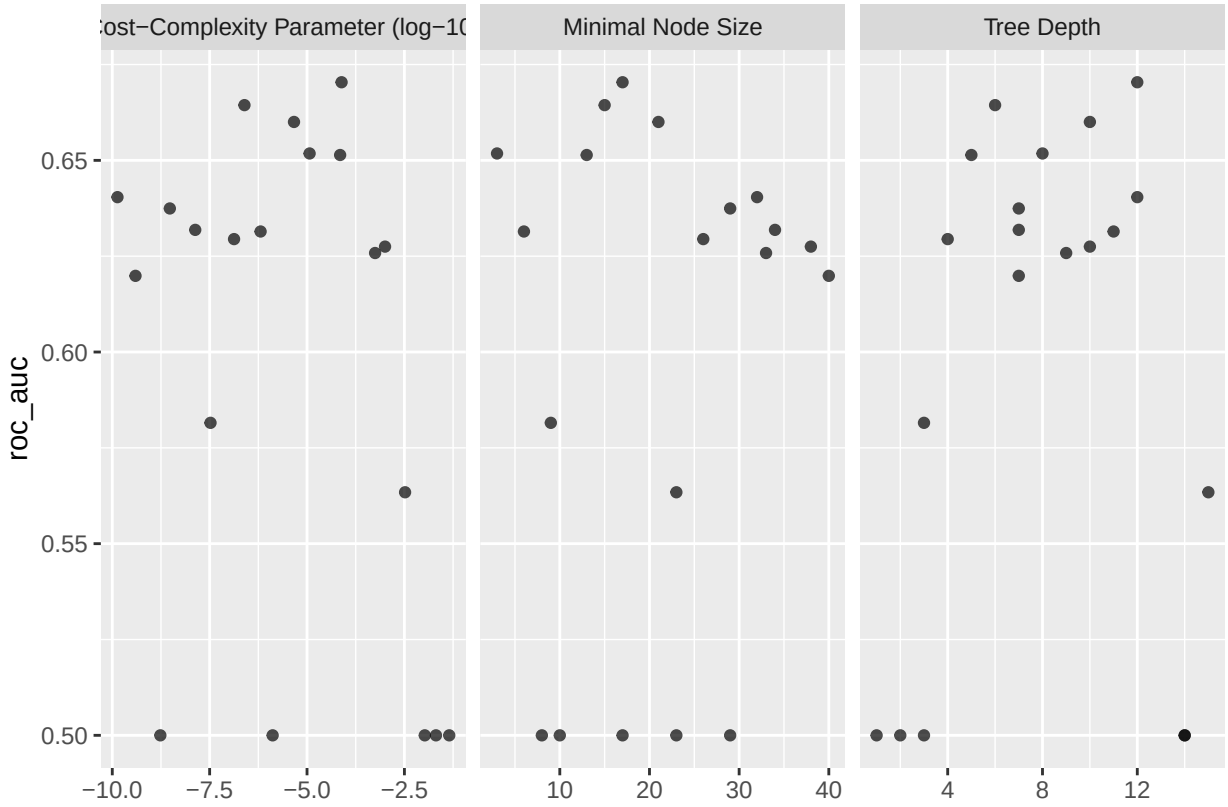
Decision Tree

```
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        3.95e-10          7    40 accuracy binary     0.952    10 0.00138 Preprocessor1_Model01
##  2        3.95e-10          7    40 roc_auc  binary     0.620    10 0.0152  Preprocessor1_Model01
##  3        1.35e- 8          7    34 accuracy binary     0.952    10 0.00142 Preprocessor1_Model02
##  4        1.35e- 8          7    34 roc_auc  binary     0.632    10 0.00744 Preprocessor1_Model02
##  5        1.31e- 6          1    17 accuracy binary     0.954    10 0.00132 Preprocessor1_Model03
##  6        1.31e- 6          1    17 roc_auc  binary     0.5      10 0       Preprocessor1_Model03
##  7        7.70e- 5         12    17 accuracy binary     0.942    10 0.00201 Preprocessor1_Model04
##  8        7.70e- 5         12    17 roc_auc  binary     0.670    10 0.0122  Preprocessor1_Model04
##  9        3.29e- 3         15    23 accuracy binary     0.952    10 0.00163 Preprocessor1_Model05
## 10        3.29e- 3         15    23 roc_auc  binary     0.563    10 0.0225  Preprocessor1_Model05
## # i 30 more rows
```

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

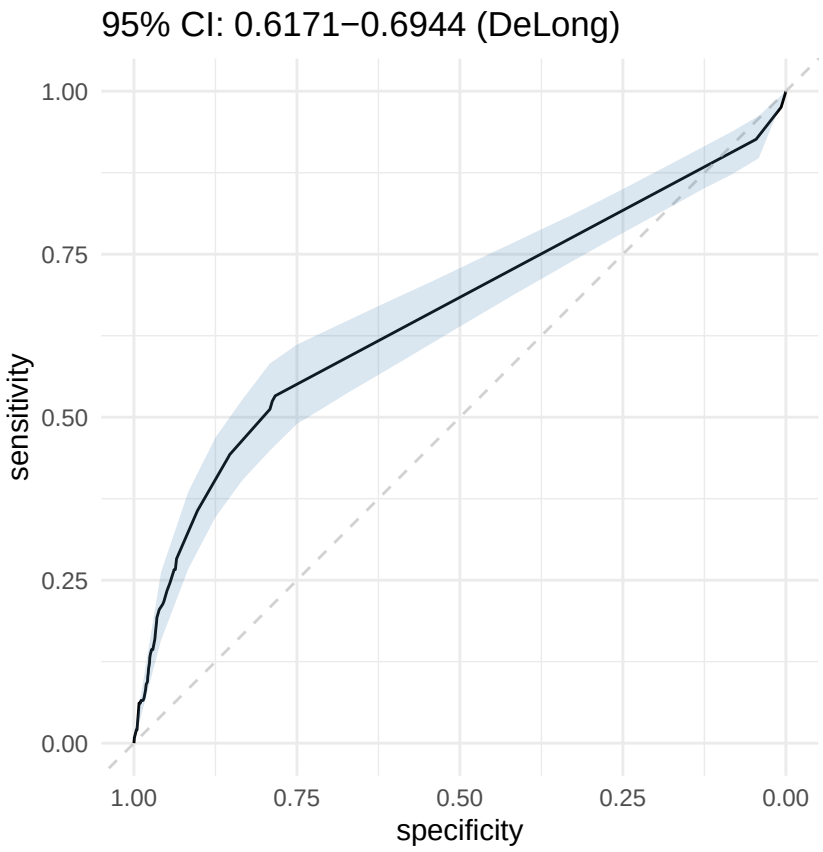



```
tree_tune %>%
  show_best("roc_auc")
```

```
## # 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     0.0000770           12    17 roc_auc binary     0.670    10  0.0122 Preprocessor1_Model04
## 2     0.000000248          6    15 roc_auc binary     0.664    10  0.0158 Preprocessor1_Model06
## 3     0.00000463          10    21 roc_auc binary     0.660    10  0.0162 Preprocessor1_Model16
## 4     0.0000116            8     3 roc_auc binary     0.652    10  0.0142 Preprocessor1_Model07
## 5     0.0000707            5    13 roc_auc binary     0.651    10  0.0140 Preprocessor1_Model20
```

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

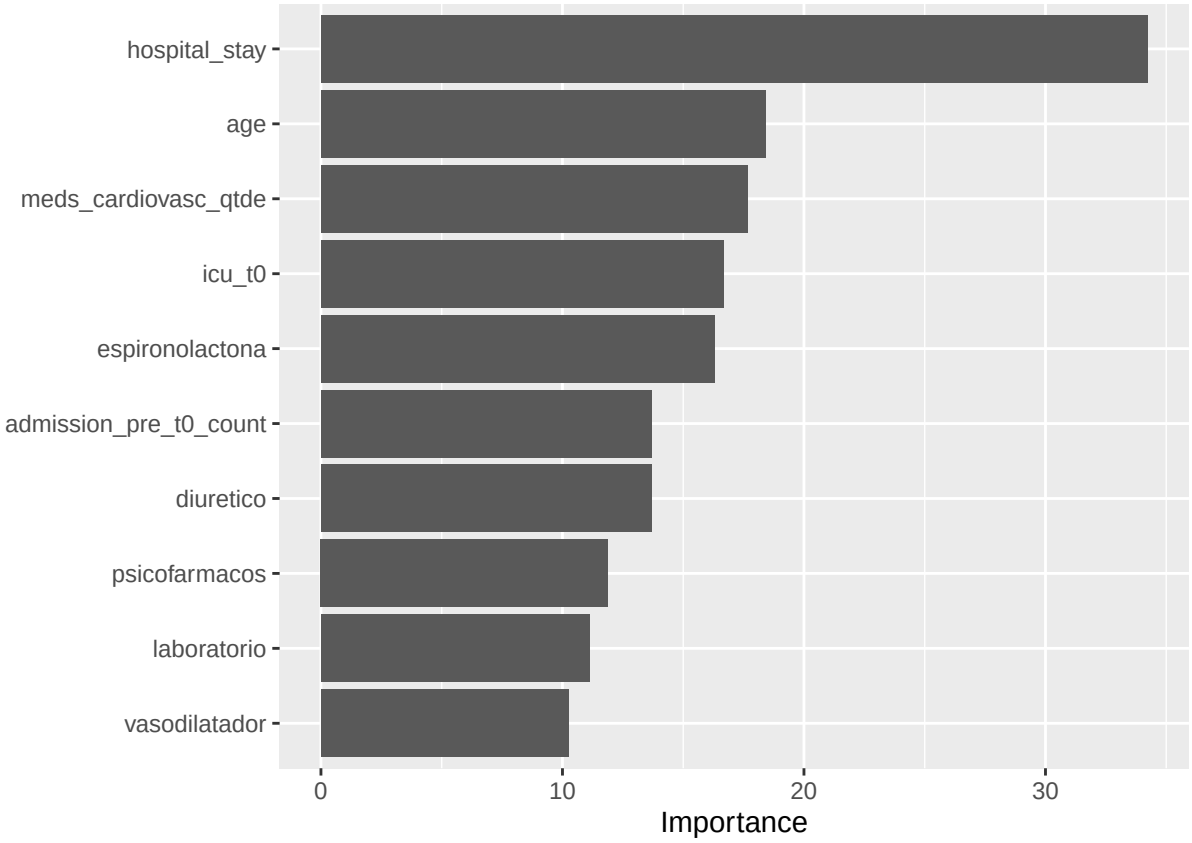
```r
final_tree_workflow <-
  tree_workflow %>%
  finalize_workflow(best_tree)

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 3514  114
##    1  972  130
##
##                Accuracy : 0.7704
##                  95% CI : (0.7581, 0.7823)
##     No Information Rate : 0.9484
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1187
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7833
##             Specificity : 0.5328
##          Pos Pred Value : 0.9686
##          Neg Pred Value : 0.1180
```

```
##              Prevalence : 0.9484
##          Detection Rate : 0.7429
##    Detection Prevalence : 0.7670
##       Balanced Accuracy : 0.6581
##
##        '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: 3.816

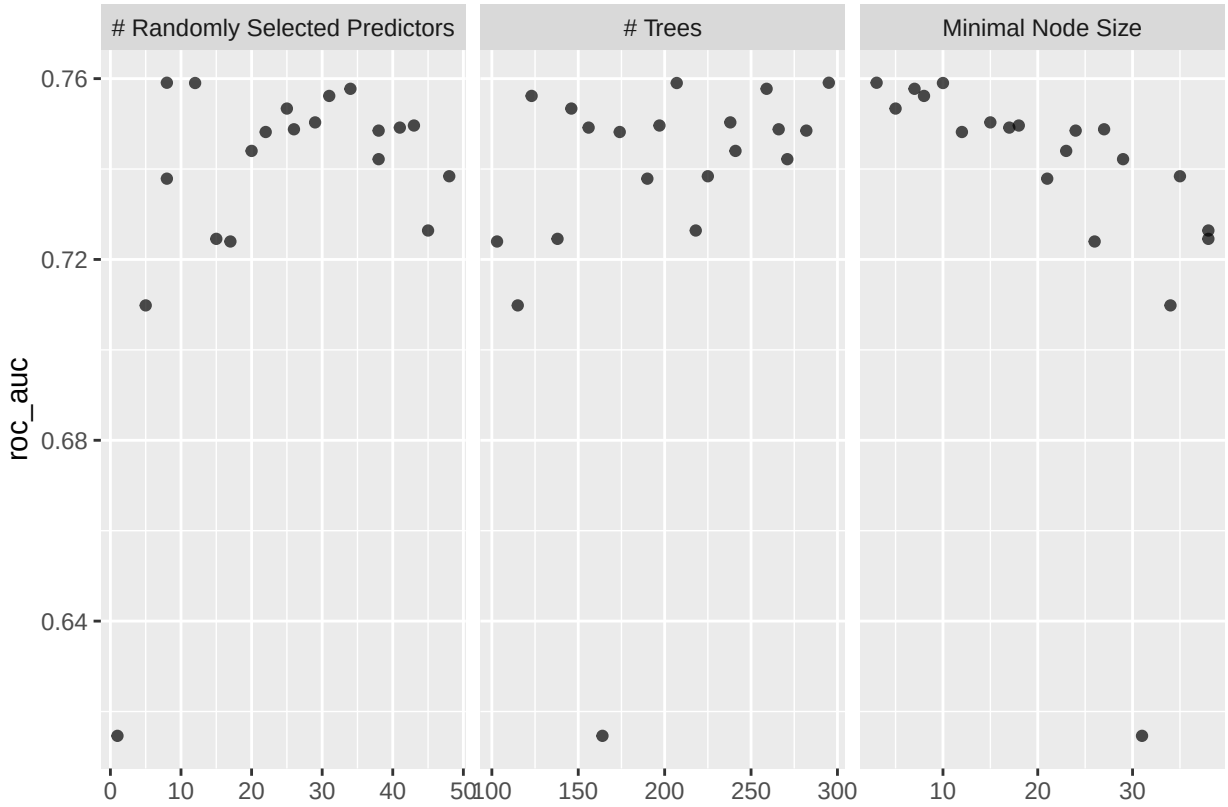
Random Forest

```
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    38   282    24 accuracy binary     0.953    10 0.00136 Preprocessor1_Model01
##  2    38   282    24 roc_auc  binary     0.748    10 0.00775 Preprocessor1_Model01
##  3     8   190    21 accuracy binary     0.954    10 0.00132 Preprocessor1_Model02
##  4     8   190    21 roc_auc  binary     0.738    10 0.00982 Preprocessor1_Model02
##  5    48   225    35 accuracy binary     0.954    10 0.00136 Preprocessor1_Model03
##  6    48   225    35 roc_auc  binary     0.738    10 0.00960 Preprocessor1_Model03
##  7     8   295     3 accuracy binary     0.954    10 0.00135 Preprocessor1_Model04
##  8     8   295     3 roc_auc  binary     0.759    10 0.00790 Preprocessor1_Model04
##  9    22   174    12 accuracy binary     0.954    10 0.00124 Preprocessor1_Model05
## 10    22   174    12 roc_auc  binary     0.748    10 0.00960 Preprocessor1_Model05
## # i 30 more rows
```

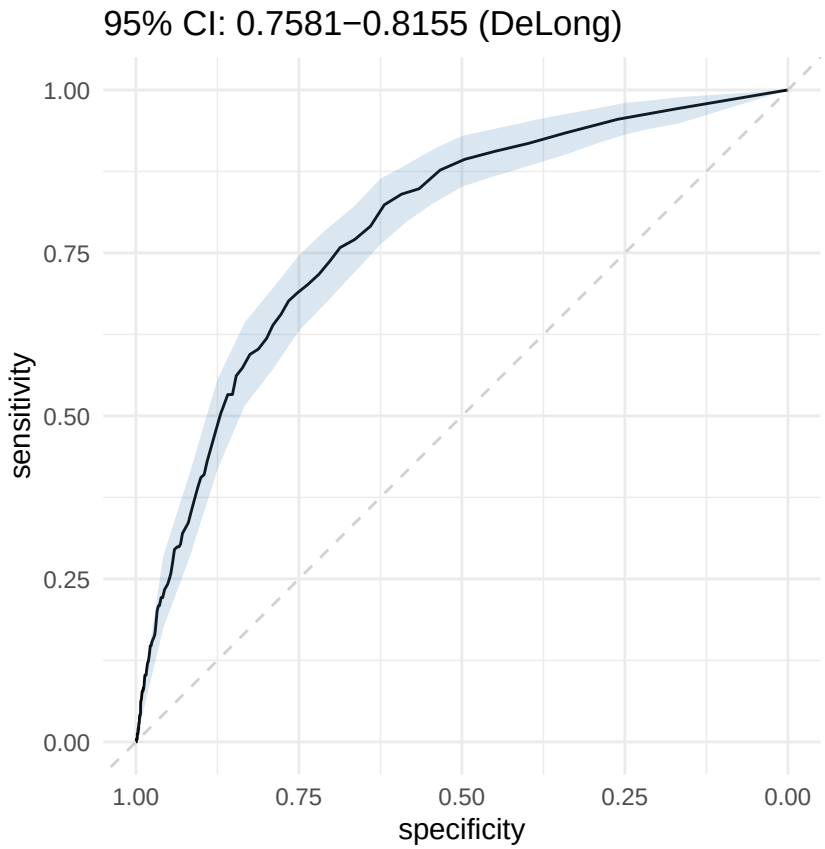
```
autoplot(rf_tune, metric = "roc_auc")
```




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

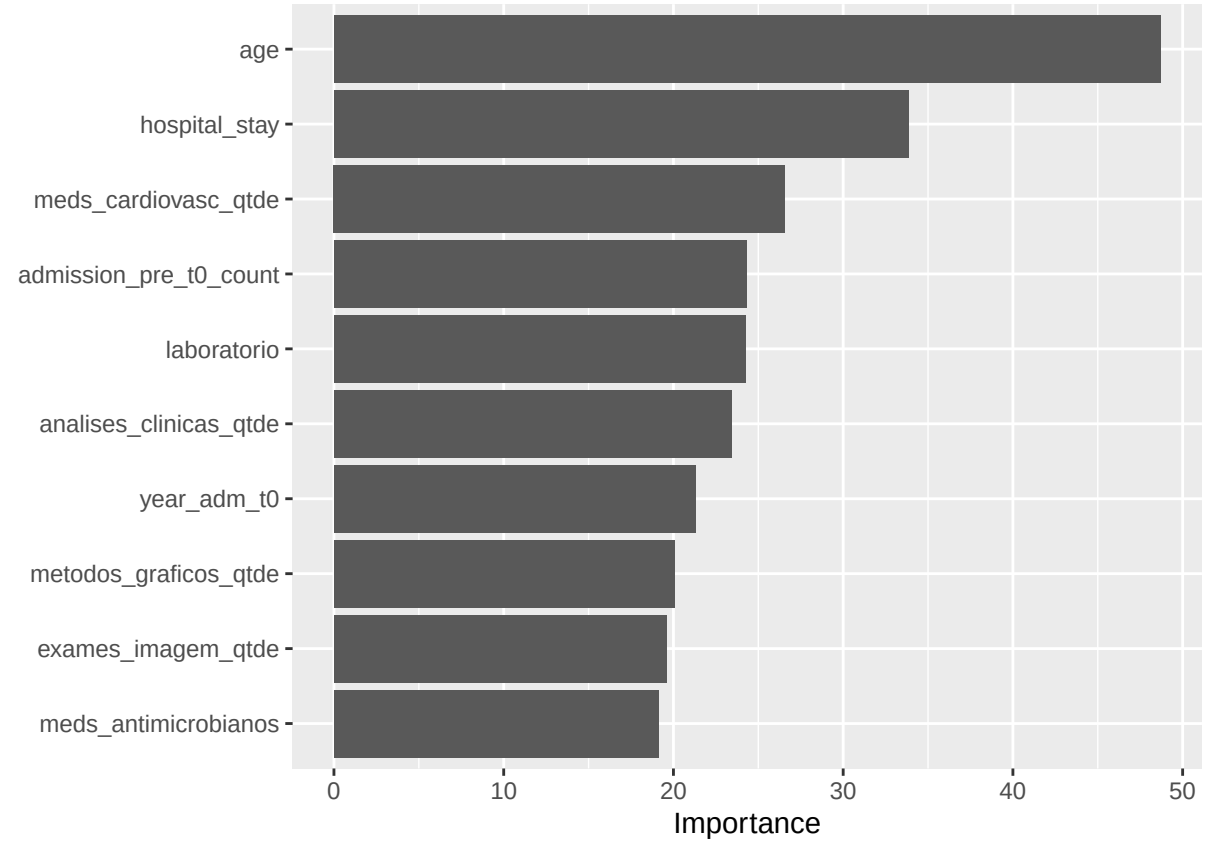
```
## # A tibble: 5 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     8   295     3 roc_auc binary     0.759    10 0.00790 Preprocessor1_Model04
## 2    12   207    10 roc_auc binary     0.759    10 0.0111  Preprocessor1_Model08
## 3    34   259     7 roc_auc binary     0.758    10 0.00995 Preprocessor1_Model18
## 4    31   123     8 roc_auc binary     0.756    10 0.00984 Preprocessor1_Model20
## 5    25   146     5 roc_auc binary     0.753    10 0.00982 Preprocessor1_Model13
```

```
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 3081   59
##    1 1405  185
##
##                Accuracy : 0.6905
##                  95% CI : (0.6771, 0.7036)
##     No Information Rate : 0.9484
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1233
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6868
##             Specificity : 0.7582
##          Pos Pred Value : 0.9812
##          Neg Pred Value : 0.1164
##              Prevalence : 0.9484
##          Detection Rate : 0.6514
##    Detection Prevalence : 0.6638
##       Balanced Accuracy : 0.7225
##
##        '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')
```

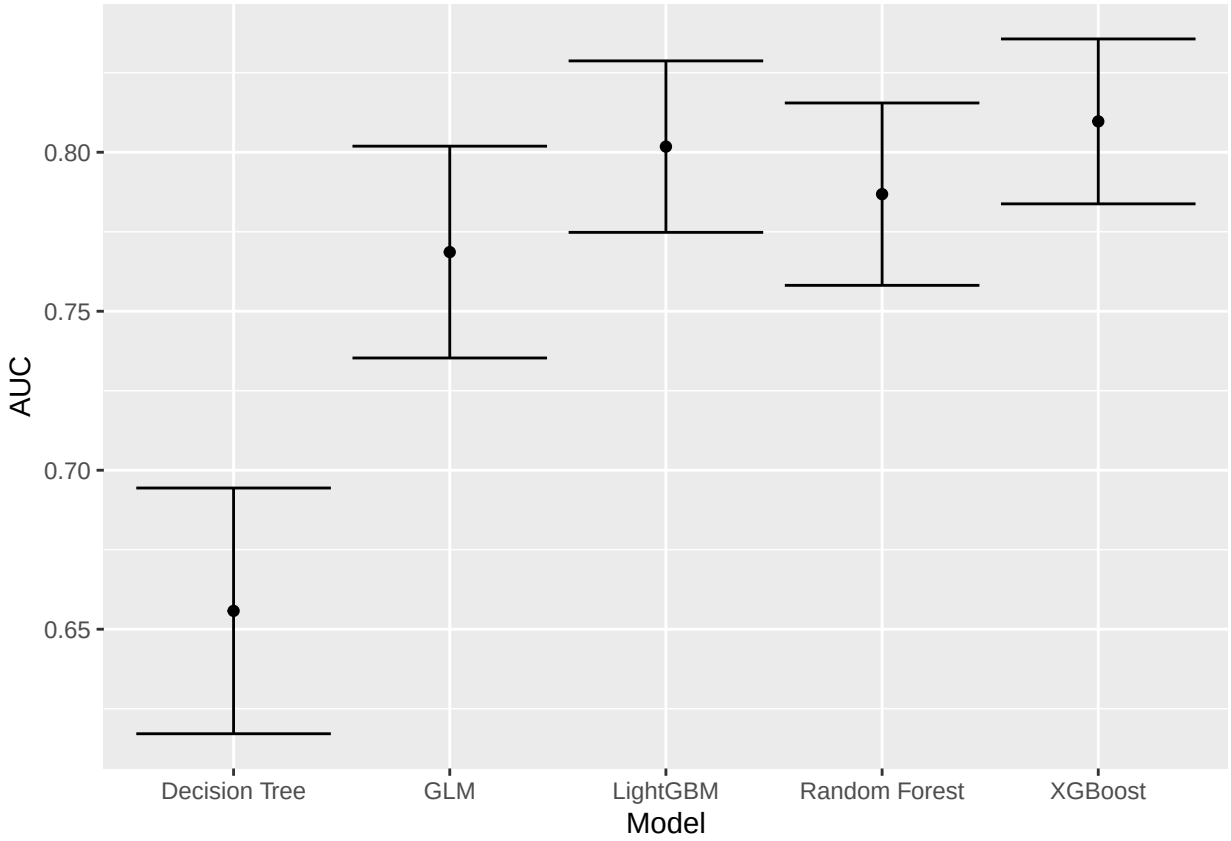



```
# extract_vip(final_rf_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 57.745

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: 0.003