

# Model Selection - death\_3year

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_3year
## . 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. race
## 04. education_level
## 05. underlying_heart_disease
## 06. heart_disease
## 07. nyha_basal
## 08. hypertension
## 09. prior_mi
## 10. heart_failure
## 11. af
## 12. cardiac_arrest
## 13. valvopathy

```

## 14. diabetes  
## 15. renal\_failure  
## 16. hemodialysis  
## 17. stroke  
## 18. copd  
## 19. comorbidities\_count  
## 20. procedure\_type\_1  
## 21. reop\_type\_1  
## 22. procedure\_type\_new  
## 23. cied\_final\_1  
## 24. cied\_final\_group\_1  
## 25. admission\_pre\_t0\_count  
## 26. admission\_pre\_t0\_180d  
## 27. year\_adm\_t0  
## 28. icu\_t0  
## 29. dialysis\_t0  
## 30. admission\_t0\_emergency  
## 31. aco  
## 32. antiarritmico  
## 33. ieca\_bra  
## 34. dva  
## 35. digoxina  
## 36. estatina  
## 37. diuretico  
## 38. vasodilatador  
## 39. insuf\_cardiaca  
## 40. espirolactona  
## 41. antiplaquetario\_ev  
## 42. insulina  
## 43. anticonvulsivante  
## 44. psicofarmacos  
## 45. antifungico  
## 46. classe\_meds\_qtde  
## 47. meds\_cardiovasc\_qtde  
## 48. meds\_antimicrobianos  
## 49. ventilacao\_mecanica  
## 50. transplante\_cardiaco  
## 51. outros\_proced\_cirurgicos  
## 52. icp  
## 53. angioplastia  
## 54. cateterismo  
## 55. eletrofisiologia  
## 56. cateter\_venoso\_central  
## 57. proced\_invasivos\_qtde  
## 58. transfusao  
## 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. angiografia  
## 71. cintilografia  
## 72. ecocardiograma  
## 73. endoscopia  
## 74. flebografia

```
## 75. pet_ct
## 76. ultrassom
## 77. tomografia
## 78. ressonancia
## 79. exames_imagem_qtde
## 80. bic
## 81. 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)
```

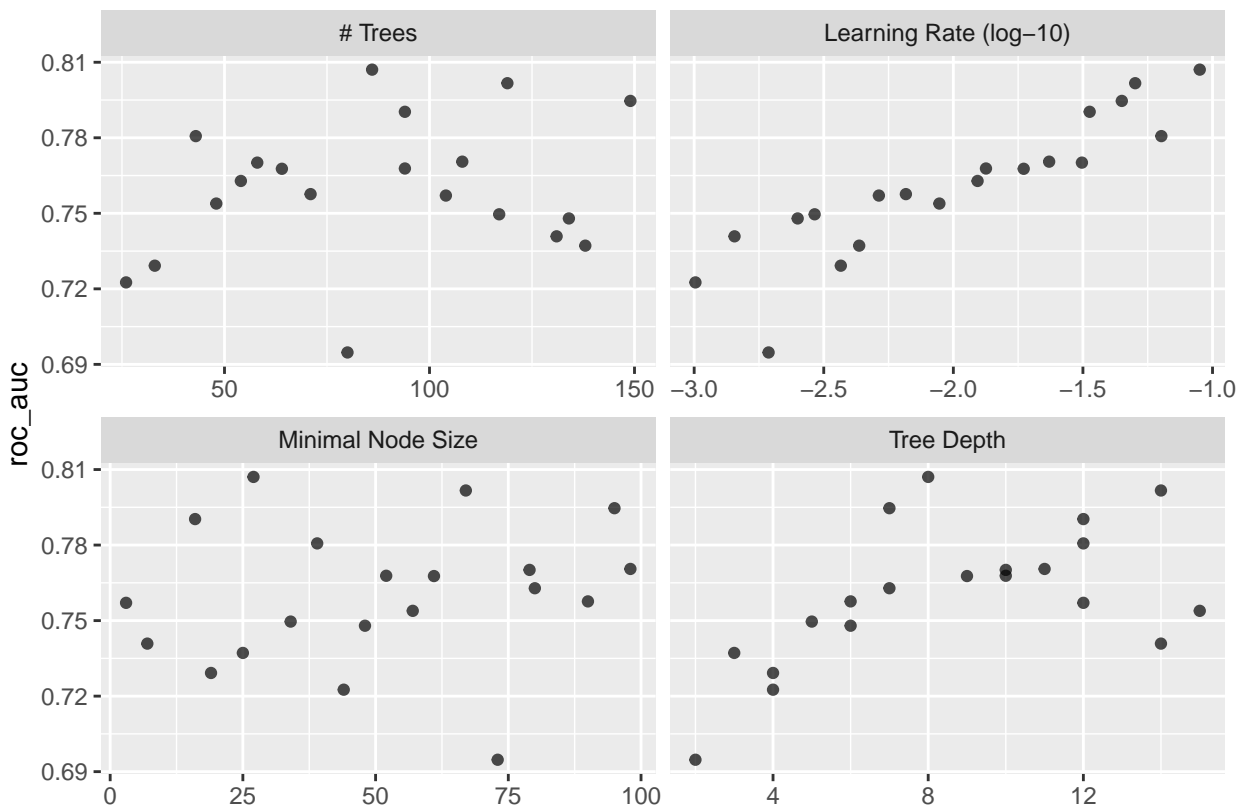
```
xgboost_tune <-
  xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)
```

```
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    86    27         8    0.0893 roc_auc binary    0.807    10 0.00914 Preprocessor1_Model06
## 2   119    67        14    0.0503 roc_auc binary    0.802    10 0.0103  Preprocessor1_Model14
## 3   149    95         7    0.0447 roc_auc binary    0.795    10 0.0104  Preprocessor1_Model19
## 4    94    16        12    0.0336 roc_auc binary    0.790    10 0.00978 Preprocessor1_Model03
## 5    43    39        12    0.0635 roc_auc binary    0.781    10 0.0106  Preprocessor1_Model08
```

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

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

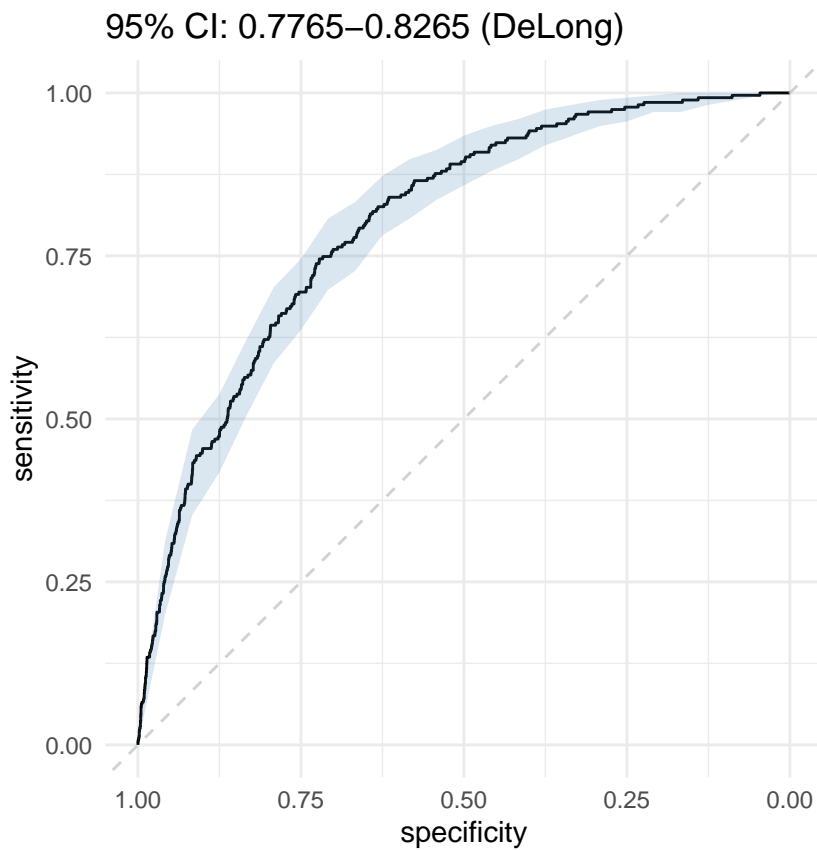


```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)
```

```
last_xgboost_fit <-
  final_xgboost_workflow %>%
  last_fit(df_split)
```

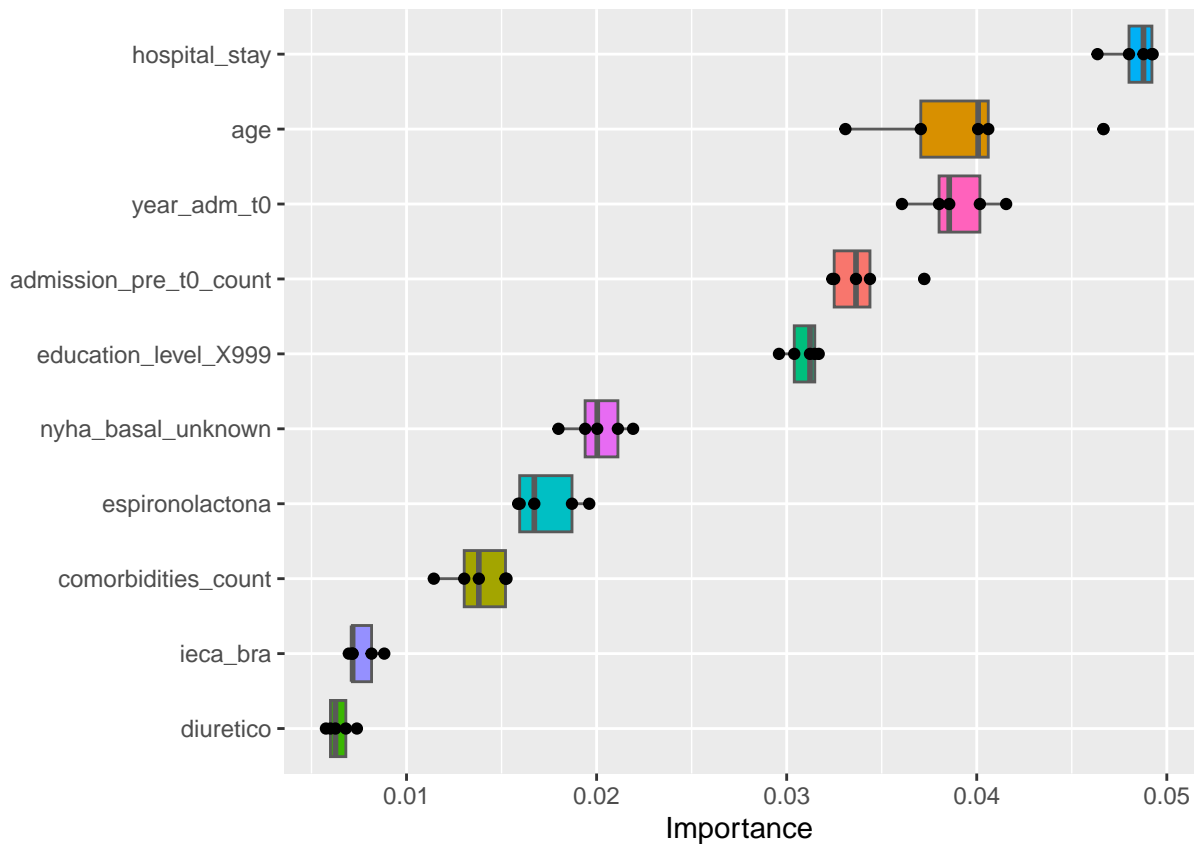
```
final_xgboost_fit <- extract_workflow(last_xgboost_fit)
```

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



```
## [1] "Optimal Threshold: 0.05"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 3216   70
##    1 1239  205
##
##              Accuracy : 0.7233
##              95% CI   : (0.7103, 0.736)
##    No Information Rate : 0.9419
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1561
##
## Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.7219
##              Specificity : 0.7455
##              Pos Pred Value : 0.9787
##              Neg Pred Value : 0.1420
##              Prevalence : 0.9419
##              Detection Rate : 0.6799
##              Detection Prevalence : 0.6947
##              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: 4.509

## 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)
```

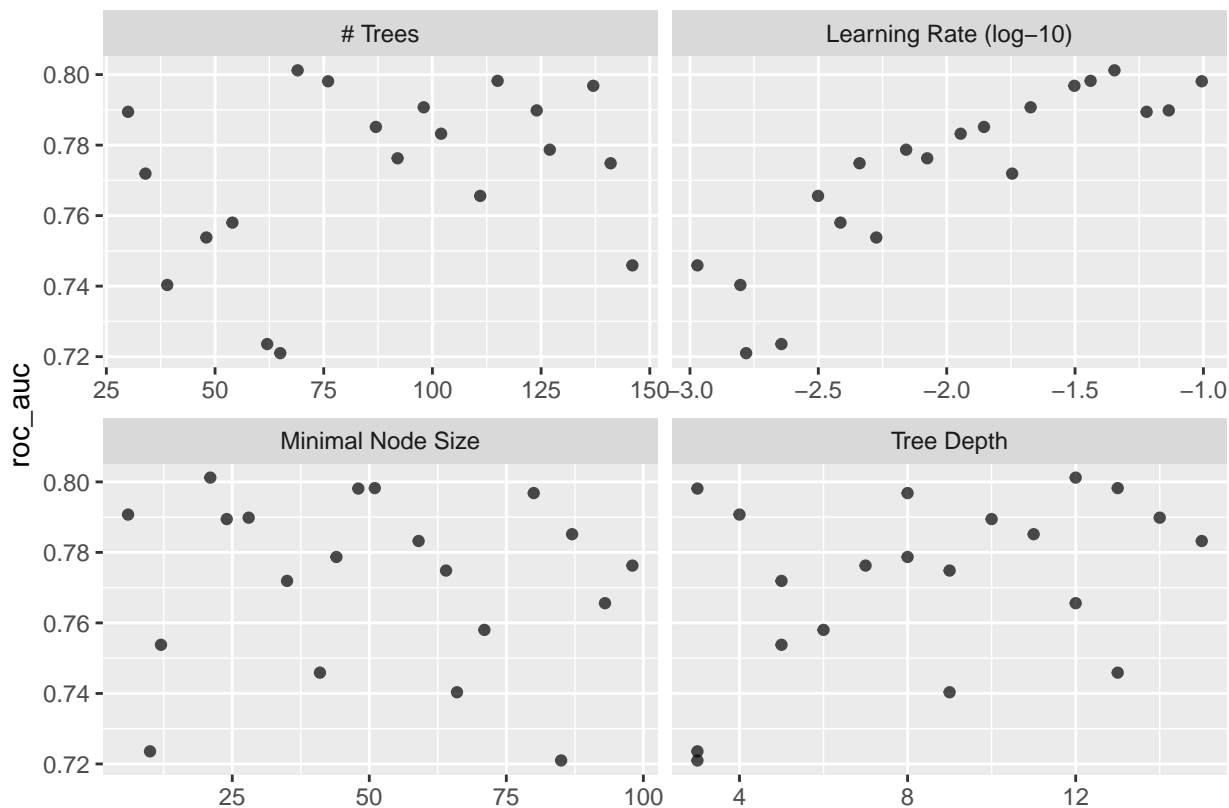
```
lightgbm_tune %>%  
  show_best("roc_auc")
```

```
## # 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    69    21      12    0.0450 roc_auc binary    0.801    10 0.00920 Preprocessor1_Model104  
## 2   115    51      13    0.0364 roc_auc binary    0.798    10 0.00971 Preprocessor1_Model111  
## 3    76    48       3    0.0985 roc_auc binary    0.798    10 0.0107  Preprocessor1_Model110  
## 4   137    80       8    0.0314 roc_auc binary    0.797    10 0.0101  Preprocessor1_Model116  
## 5    98     6       4    0.0212 roc_auc binary    0.791    10 0.0102  Preprocessor1_Model101
```

```
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 <-
```



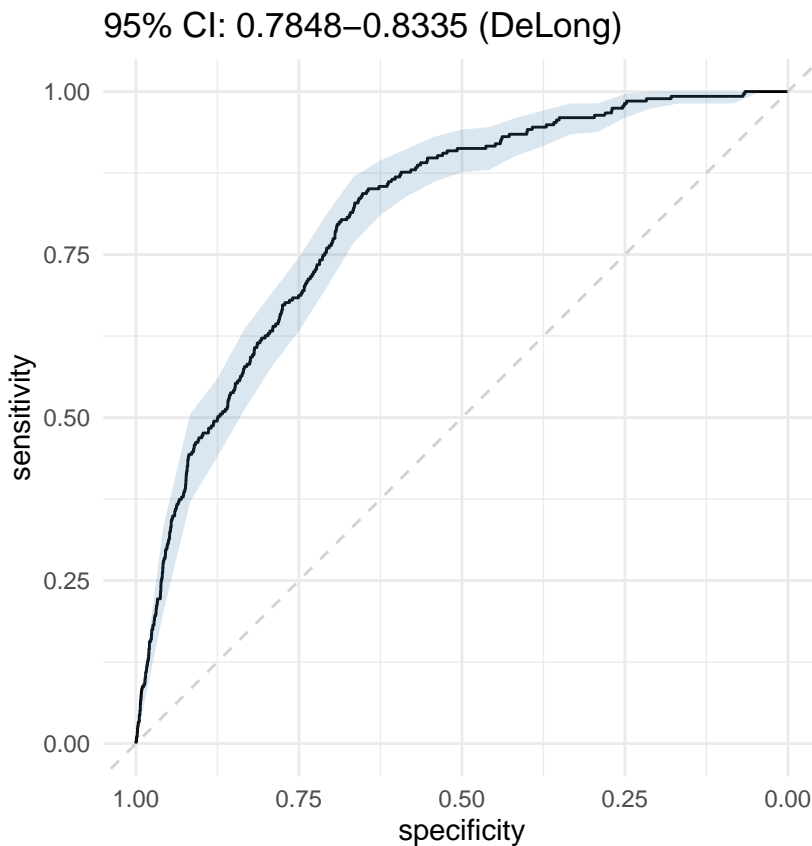
```

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: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 2907  43
## 1 1548 232
##
##              Accuracy : 0.6636
##              95% CI : (0.65, 0.6771)
##    No Information Rate : 0.9419
##    P-Value [Acc > NIR] : 1
##
##              Kappa : 0.1391
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6525
##              Specificity : 0.8436
##    Pos Pred Value : 0.9854
##    Neg Pred Value : 0.1303
##    Prevalence : 0.9419
##    Detection Rate : 0.6146
##    Detection Prevalence : 0.6237
##    Balanced Accuracy : 0.7481
##

```

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

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

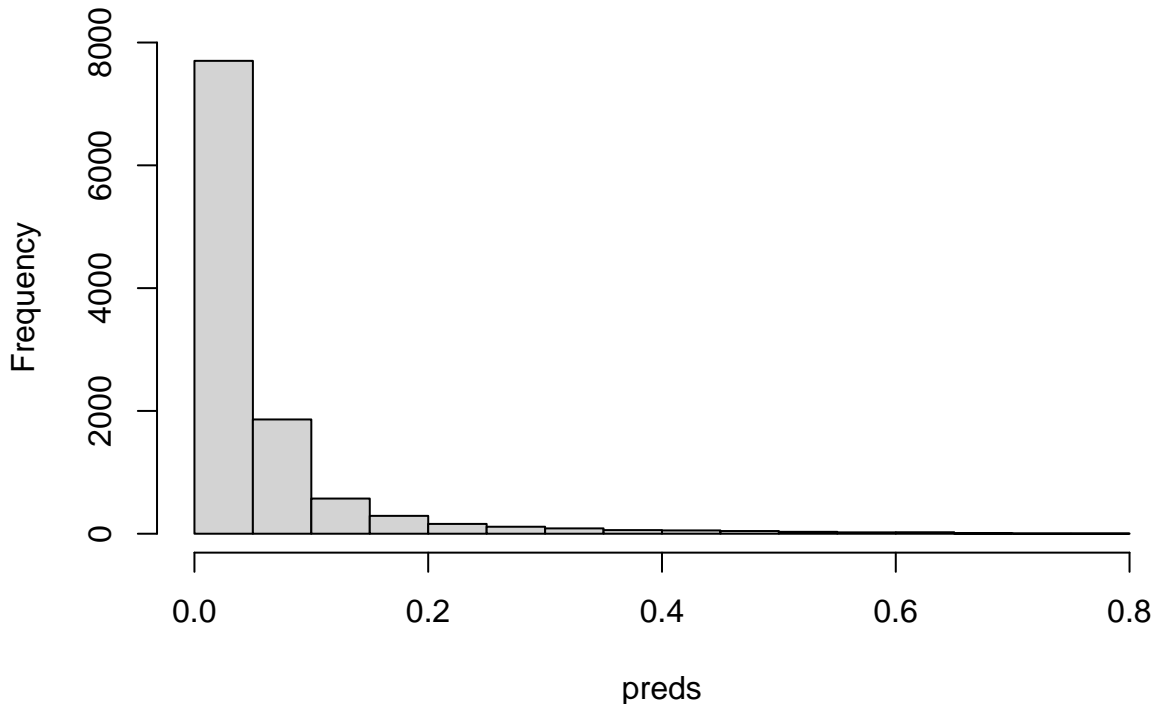
```
Hmisc::list.tree(lightgbm_parameters)
```

```
## lightgbm_parameters = list 4 (736 bytes)
## . trees = integer 1= 69
## . min_n = integer 1= 21
## . tree_depth = integer 1= 12
## . learn_rate = double 1= 0.045034
```

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

Minutes to run: 3.605

## Histogram of preds



Minutes to run:

0.008

## 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())
```

```

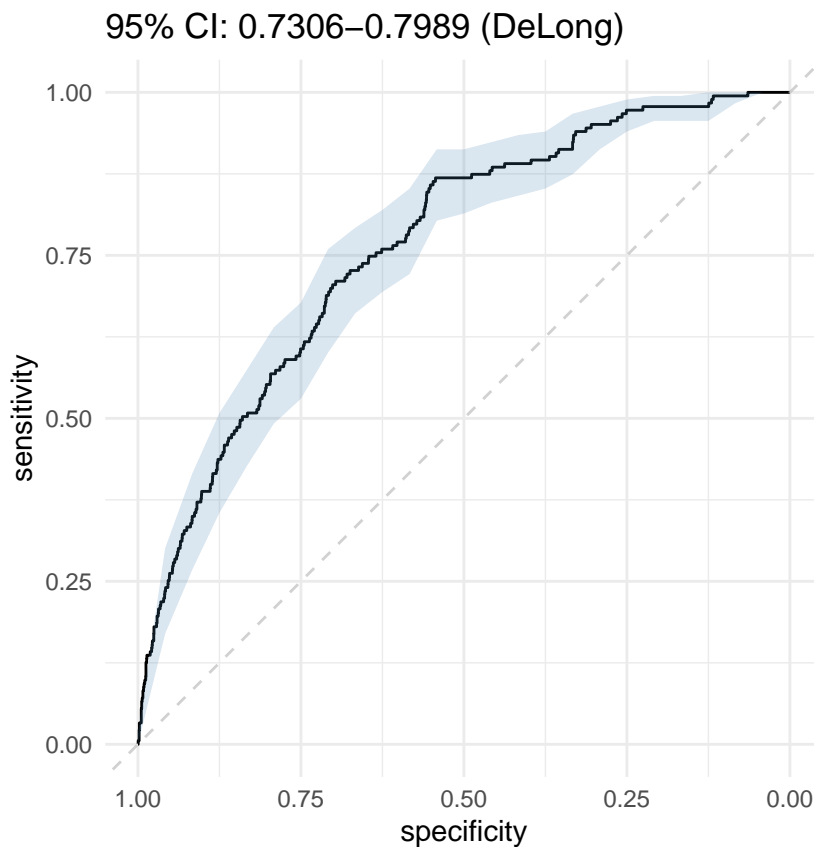
glmnet_spec <-
  logistic_reg(penalty = 0) %>%
  set_mode("classification") %>%
  set_engine("glm")

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)

```



```

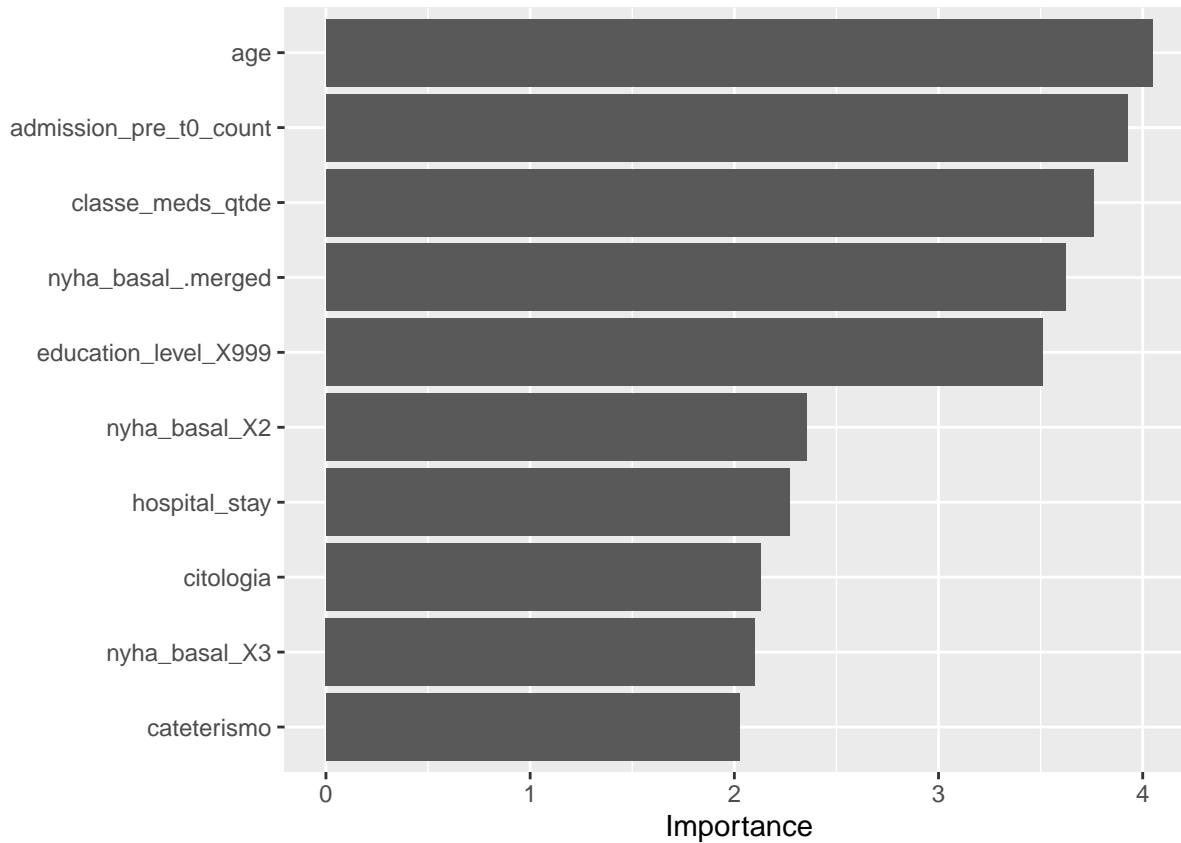
## [1] "Optimal Threshold: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 1502   24
## 1 1261  159
##
##              Accuracy : 0.5638
##              95% CI   : (0.5457, 0.5818)
##    No Information Rate : 0.9379
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.0992
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.5436

```

```
##          Specificity : 0.8689
##          Pos Pred Value : 0.9843
##          Neg Pred Value : 0.1120
##          Prevalence : 0.9379
##          Detection Rate : 0.5098
##          Detection Prevalence : 0.5180
##          Balanced Accuracy : 0.7062
##
##          '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:

0.194

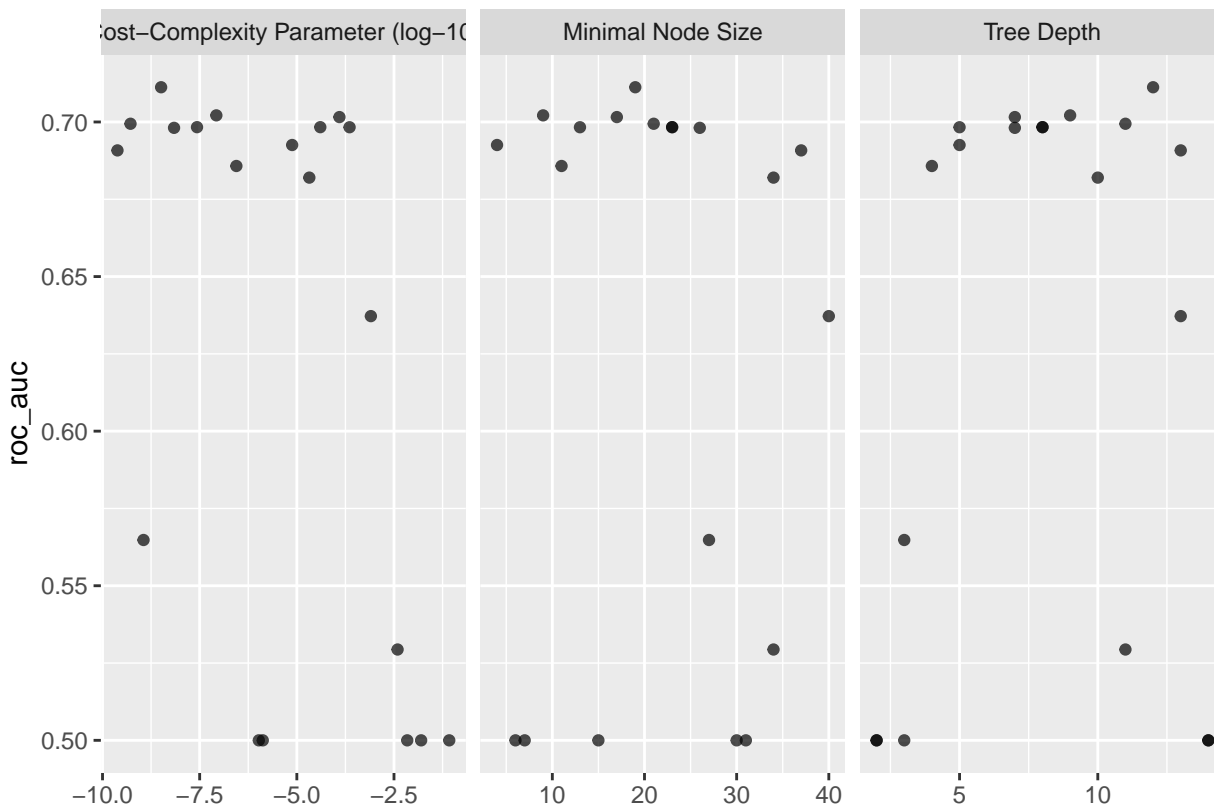
## Decision Tree

```
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  0.00693             3      15 accuracy binary   0.942    10 0.00259 Preprocessor1_Model101
## 2  0.00693             3      15 roc_auc  binary   0.5      10 0        Preprocessor1_Model101
## 3  0.00000000693       7      26 accuracy binary   0.938    10 0.00278 Preprocessor1_Model102
## 4  0.00000000693       7      26 roc_auc  binary   0.698    10 0.0160  Preprocessor1_Model102
## 5  0.000227            5      13 accuracy binary   0.939    10 0.00218 Preprocessor1_Model103
## 6  0.000227            5      13 roc_auc  binary   0.698    10 0.0157  Preprocessor1_Model103
## 7  0.0000000849        9        9 accuracy binary   0.928    10 0.00296 Preprocessor1_Model104
## 8  0.0000000849        9        9 roc_auc  binary   0.702    10 0.0143  Preprocessor1_Model104
## 9  0.0835              14        6 accuracy binary   0.942    10 0.00259 Preprocessor1_Model105
## 10 0.0835              14        6 roc_auc  binary   0.5      10 0        Preprocessor1_Model105
## # 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  3.22e- 9          12      19 roc_auc  binary   0.711    10 0.0138 Preprocessor1_Model108
## 2  8.49e- 8           9        9 roc_auc  binary   0.702    10 0.0143 Preprocessor1_Model104
## 3  1.25e- 4           7      17 roc_auc  binary   0.702    10 0.0144 Preprocessor1_Model113
## 4  5.25e-10          11      21 roc_auc  binary   0.699    10 0.0127 Preprocessor1_Model118
## 5  2.70e- 8           8      23 roc_auc  binary   0.698    10 0.0157 Preprocessor1_Model116
```

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

```

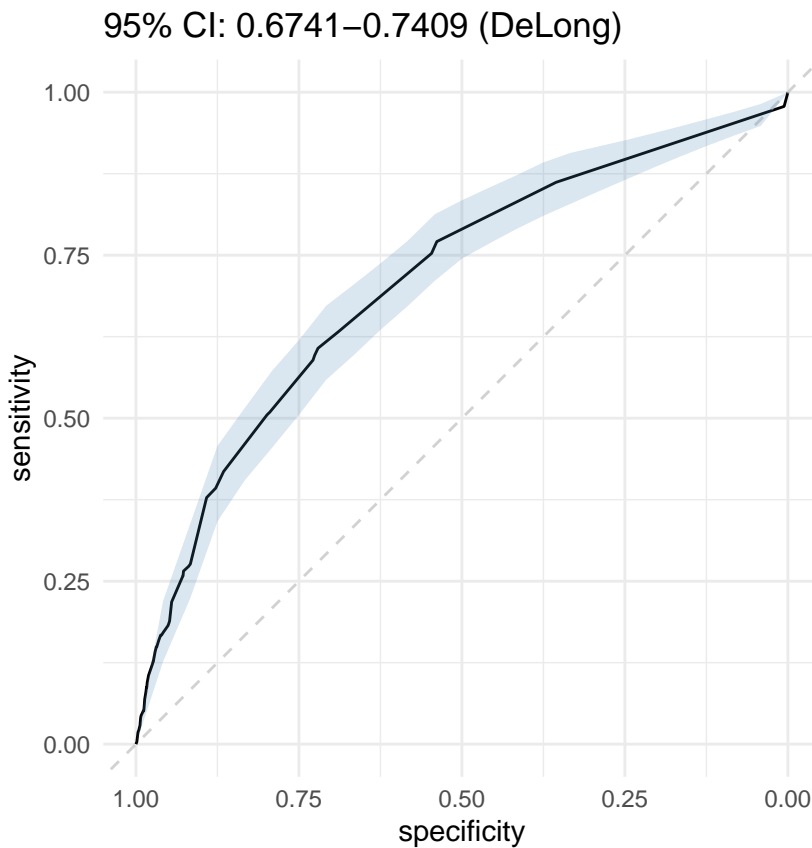
final_tree_workflow <-
  tree_workflow %>%
  finalize_workflow(best_tree)

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)

```



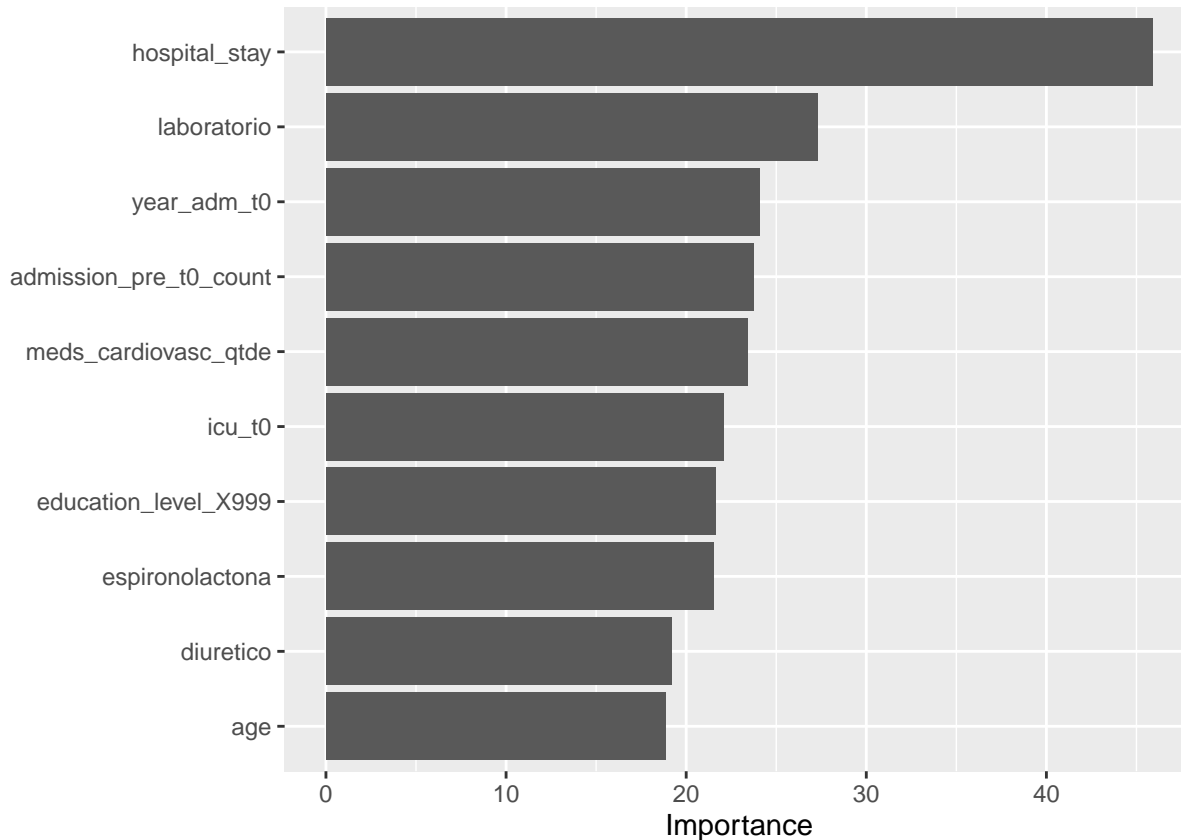
```

## [1] "Optimal Threshold: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##      0 3211 108
##      1 1244 167
##
##              Accuracy : 0.7142
##              95% CI : (0.7011, 0.727)
##      No Information Rate : 0.9419
##      P-Value [Acc > NIR] : 1
##
##              Kappa : 0.1117
##
##      McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.7208
##              Specificity : 0.6073
##      Pos Pred Value : 0.9675
##      Neg Pred Value : 0.1184

```

```
##           Prevalence : 0.9419
##           Detection Rate : 0.6789
##           Detection Prevalence : 0.7017
##           Balanced Accuracy : 0.6640
##
##           '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: 3.953

## 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)

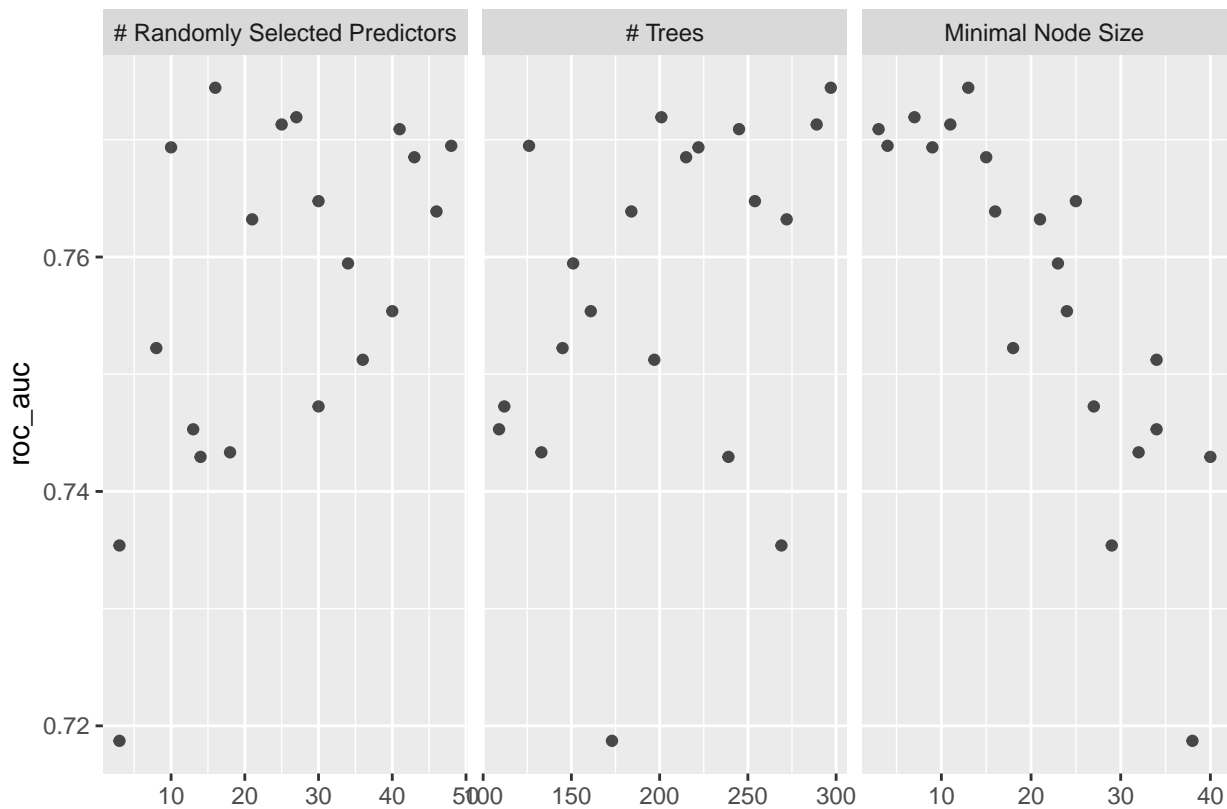
rf_tune <-
  rf_workflow %>%
  tune_grid(resamples = df_folds,
    grid = rf_grid)

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   126     4 accuracy binary    0.942    10 0.00283 Preprocessor1_Model01
## 2     48   126     4 roc_auc  binary    0.769    10 0.00964 Preprocessor1_Model01
## 3     21   272    21 accuracy binary    0.942    10 0.00259 Preprocessor1_Model02
## 4     21   272    21 roc_auc  binary    0.763    10 0.0124  Preprocessor1_Model02
## 5     25   289    11 accuracy binary    0.942    10 0.00255 Preprocessor1_Model03
## 6     25   289    11 roc_auc  binary    0.771    10 0.0107  Preprocessor1_Model03
## 7     18   133    32 accuracy binary    0.942    10 0.00259 Preprocessor1_Model04
## 8     18   133    32 roc_auc  binary    0.743    10 0.0117  Preprocessor1_Model04
## 9     30   254    25 accuracy binary    0.941    10 0.00270 Preprocessor1_Model05
## 10    30   254    25 roc_auc  binary    0.765    10 0.0123  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    16   297    13 roc_auc binary  0.774    10 0.0127 Preprocessor1_Model19
## 2    27   201     7 roc_auc binary  0.772    10 0.0110 Preprocessor1_Model09
## 3    25   289    11 roc_auc binary  0.771    10 0.0107 Preprocessor1_Model03
## 4    41   245     3 roc_auc binary  0.771    10 0.00887 Preprocessor1_Model07
## 5    48   126     4 roc_auc binary  0.769    10 0.00964 Preprocessor1_Model01
```

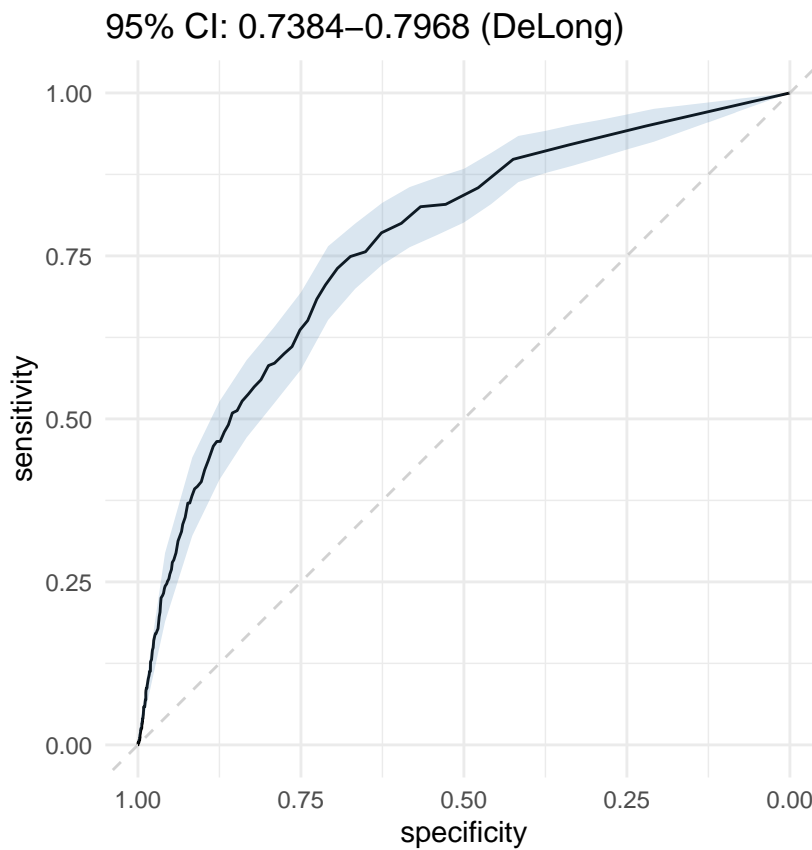
```
best_rf <- rf_tune %>%
  select_best("roc_auc")
```

```
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)
```

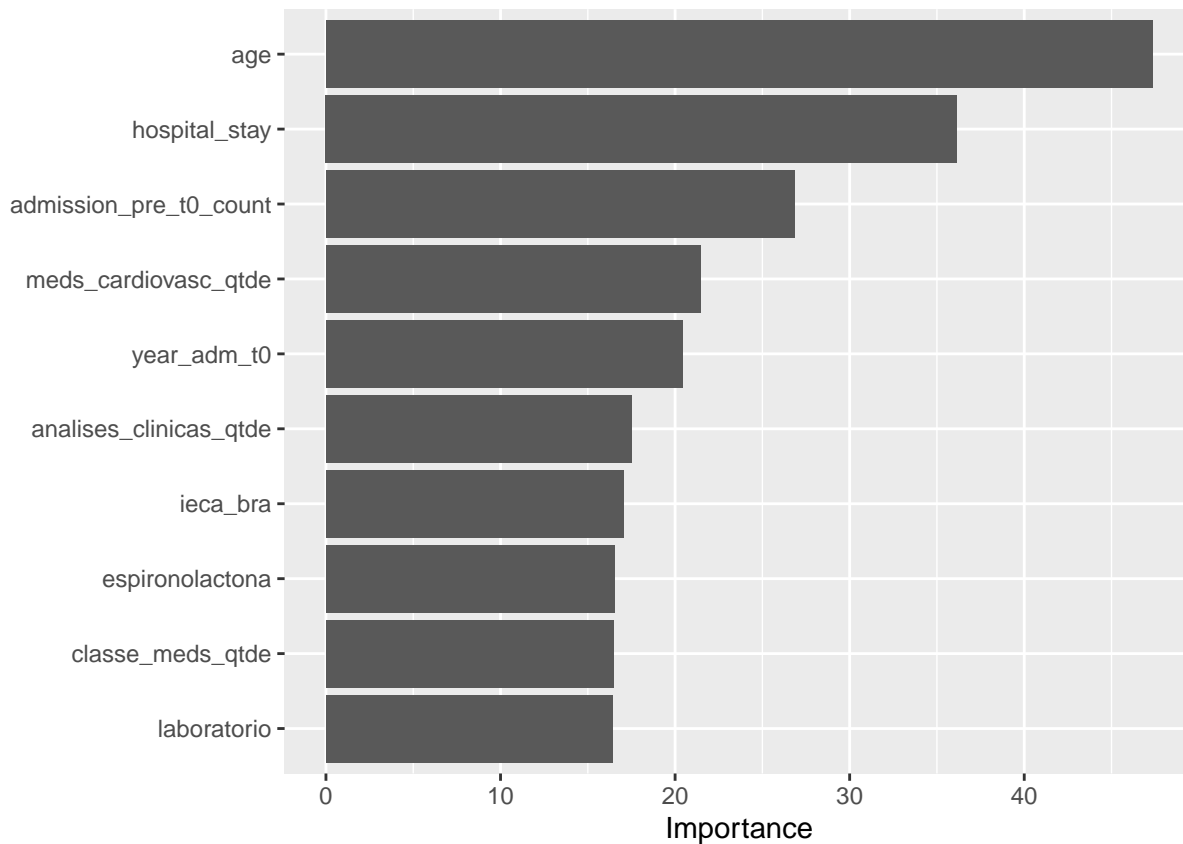
```
rf_auc <- validation(final_rf_fit, df_test)
```



```
## [1] "Optimal Threshold: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 3092   74
##    1 1363  201
##
##              Accuracy : 0.6962
##              95% CI   : (0.6829, 0.7093)
##    No Information Rate : 0.9419
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1328
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6941
##              Specificity : 0.7309
##              Pos Pred Value : 0.9766
##              Neg Pred Value : 0.1285
##              Prevalence : 0.9419
##              Detection Rate : 0.6537
##              Detection Prevalence : 0.6693
##              Balanced Accuracy : 0.7125
##
##              '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')
```



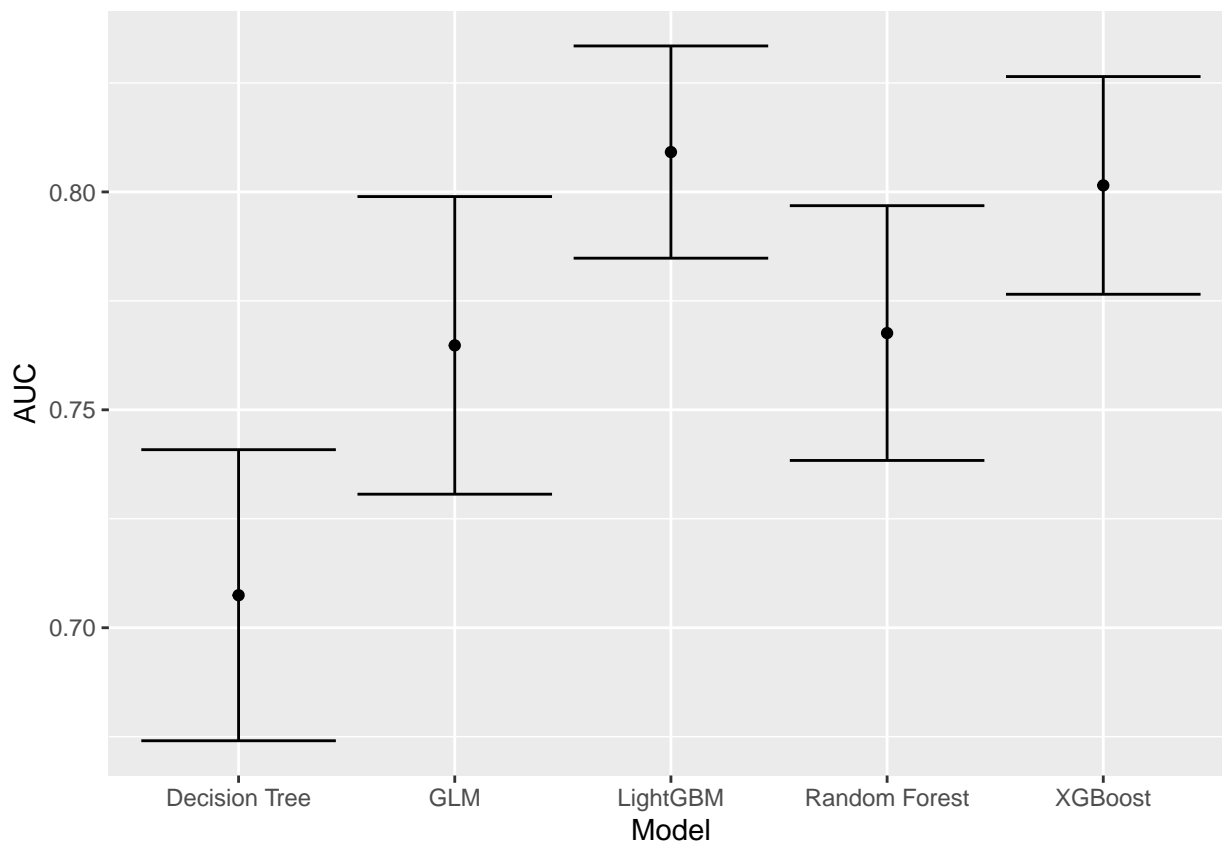
```
# extract_vip(final_rf_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 56.654

## 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