Model Selection - death_3year

Eduardo Yuki Yada

Global parameters

```r
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= 30
## .  repeats = double 1= 2
## .  RUN_ALL_MODELS = logical 1= TRUE
```

Minutes to run: 0

Imports

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

source("aux_functions.R")
```

Minutes to run: 0

Loading data

```r
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: 0.007

```r
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

```r
cat_features_list = readRDS(sprintf(
  "./auxiliar/significant_columns/categorical_%s.rds",
  outcome_column
))

num_features_list = readRDS(sprintf(
  "./auxiliar/significant_columns/numerical_%s.rds",
  outcome_column
))

features_list = c(cat_features_list, num_features_list)
```

Minutes to run: 0

```r
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. espironolactona
## 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%)

```r
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 <- rsample::testing(df_split) %>% dplyr::select(all_of(c(features, outcome_column)))

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

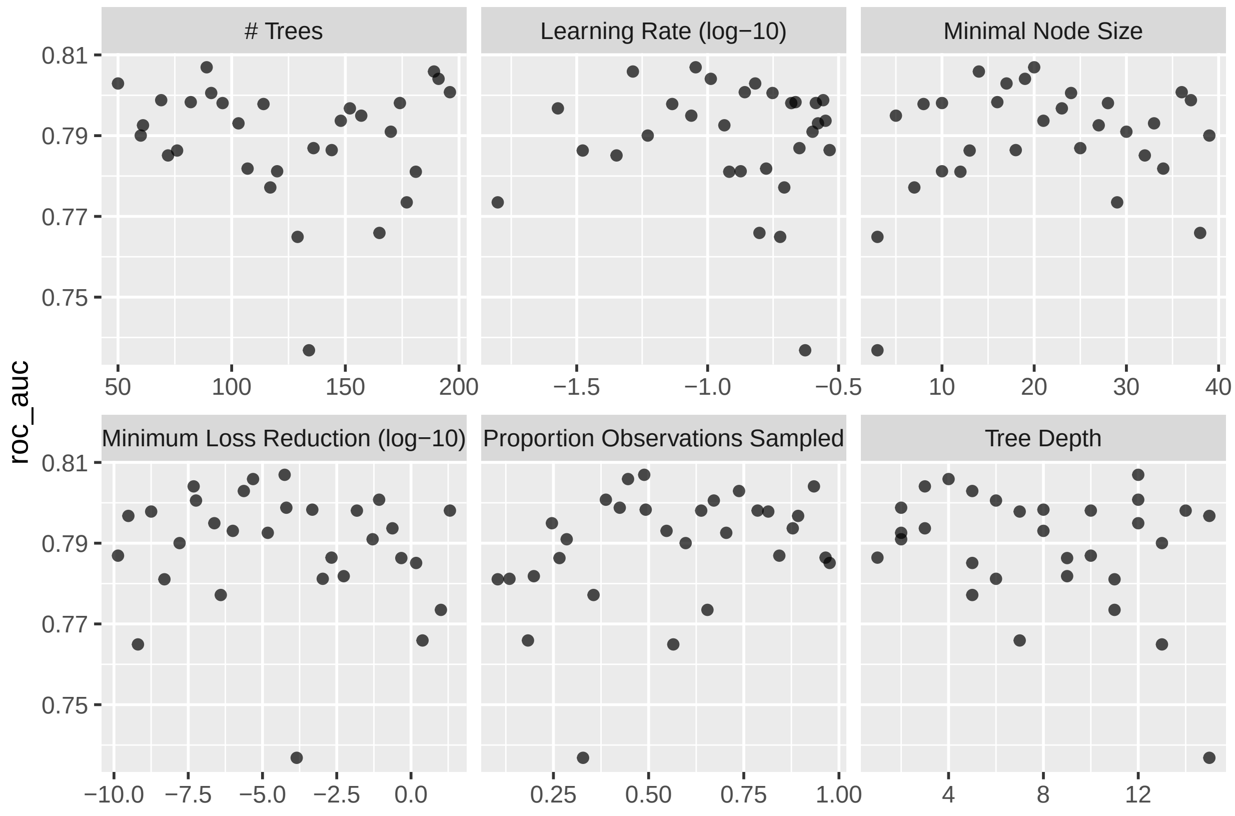
Minutes to run: 0.001

Boosted Tree (XGBoost)

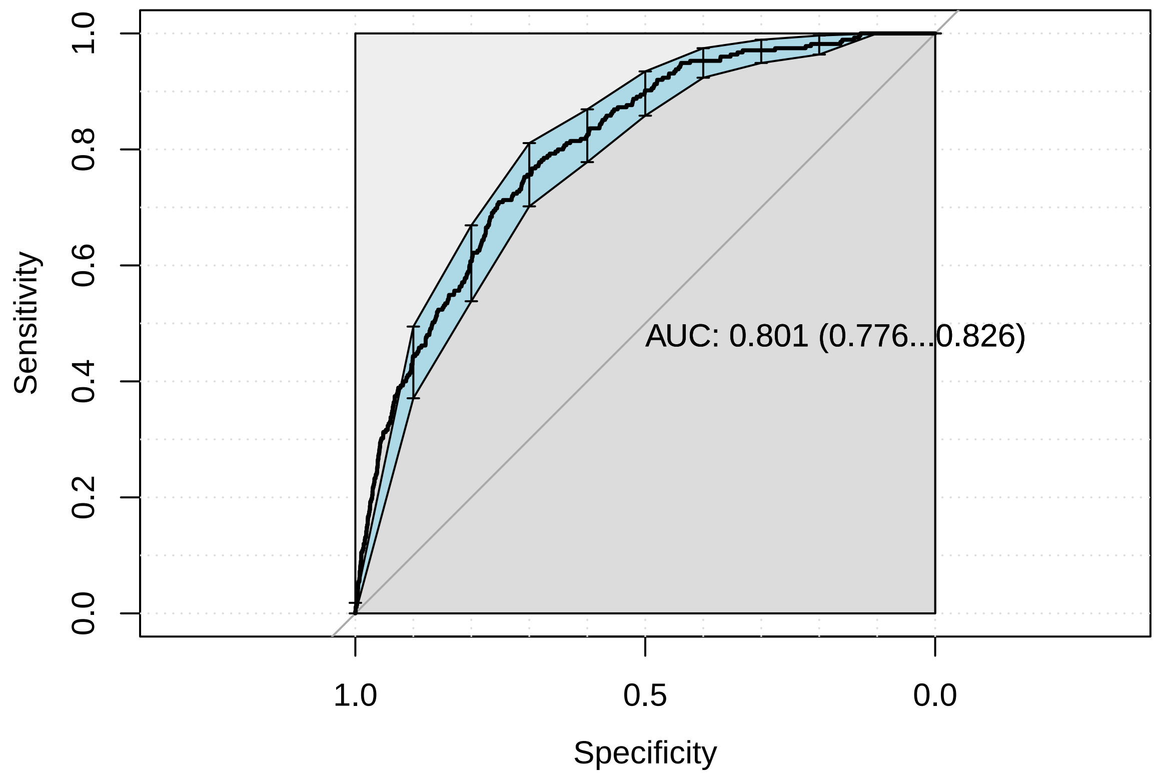
```r
xgboost_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors())

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

xgboost_grid <- grid_latin_hypercube(
  trees(range = c(50L, 200L)),
  min_n(),
  tree_depth(),
  learn_rate(range = c(0.01, 0.3), trans = NULL),
  loss_reduction(),
  sample_prop(range = c(1/10, 1), trans = NULL),
  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 12
##   trees min_n tree_depth learn_rate loss_reduction sample_size .metric .estimator  mean     n std_err .config
##   <int> <int>      <int>      <dbl>          <dbl>       <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    89    20         12     0.0900   0.0000556          0.488 roc_auc binary     0.807    10 0.0107  Preproc
## 2   189    14          4     0.0518   0.00000481         0.446 roc_auc binary     0.806    10 0.00962 Preproc
## 3   191    19          3     0.103    0.0000000481       0.934 roc_auc binary     0.804    10 0.00917 Preproc
## 4    50    17          5     0.152    0.00000235         0.738 roc_auc binary     0.803    10 0.0105  Preproc
## 5   196    36         12     0.139    0.0844             0.388 roc_auc binary     0.801    10 0.0107  Preproc
```

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

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

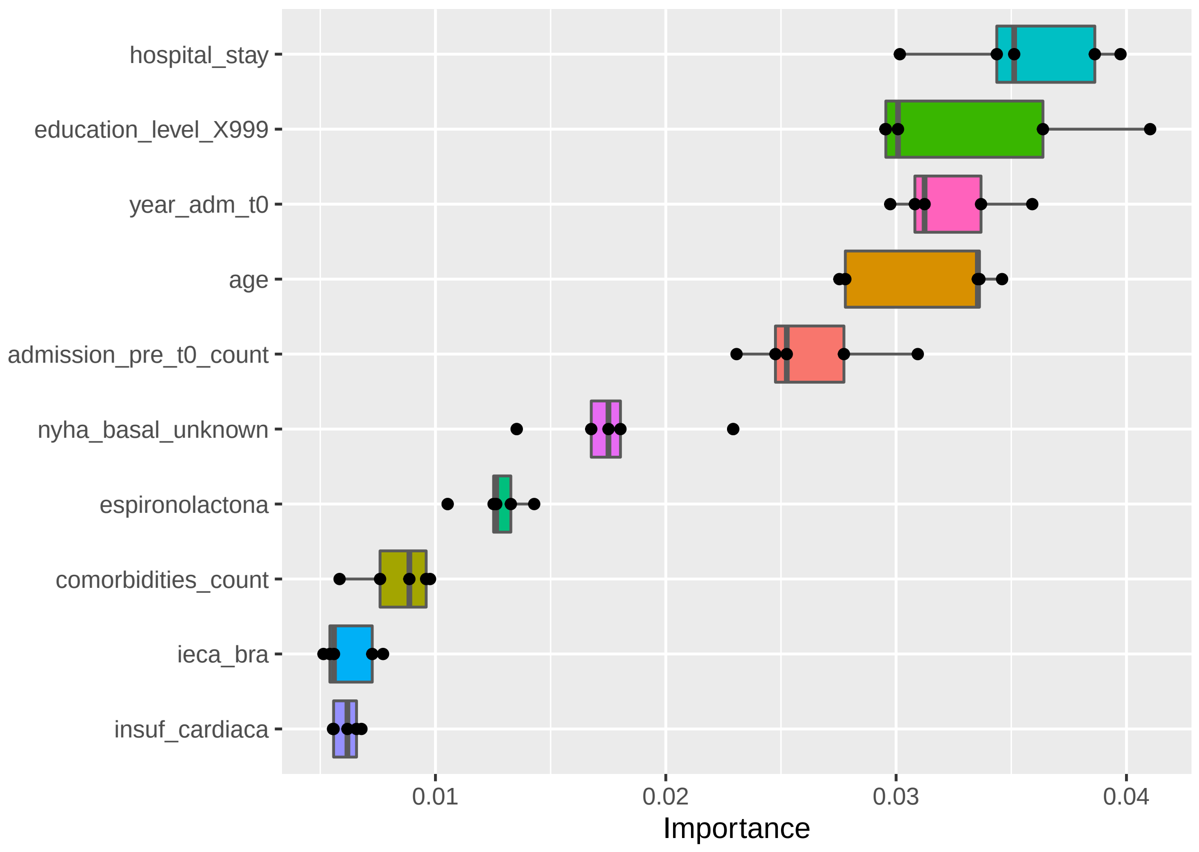


```r
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

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)
```

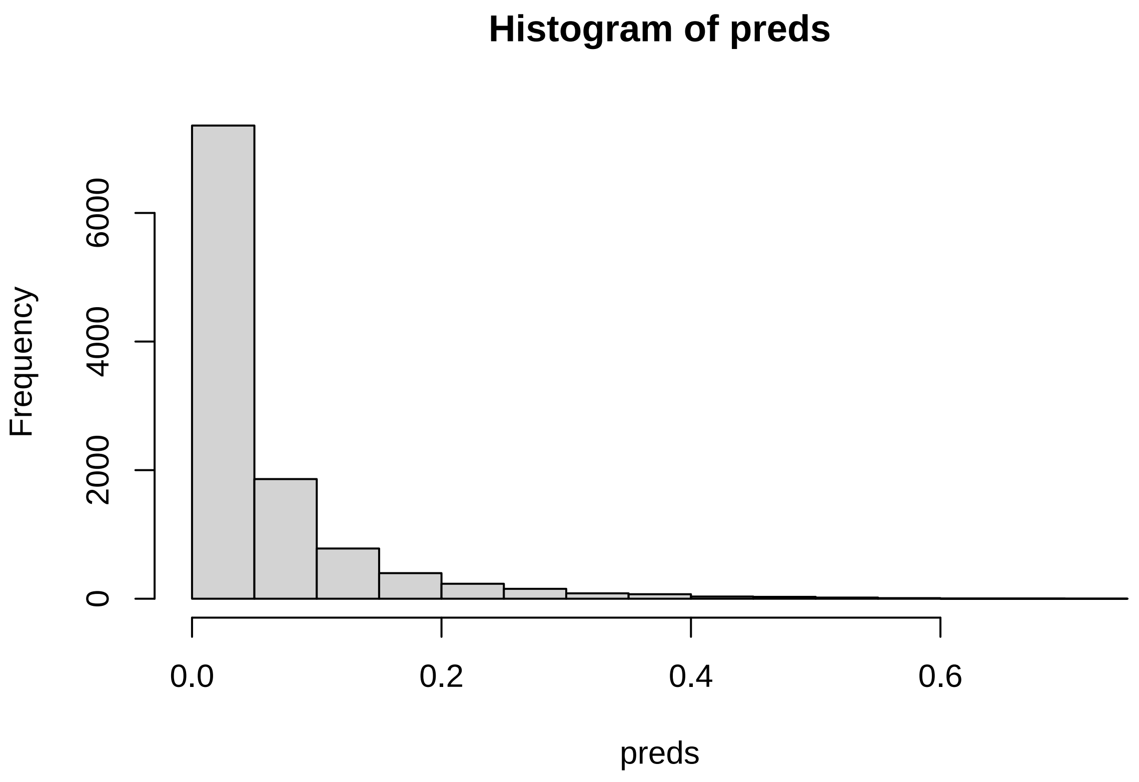


```
##   |
```

```
extract_vip(final_xgboost_fit, pred_wrapper = predict,
            reference_class = "0")
```



```
xgboost_parameters <- xgboost_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(trees, min_n, tree_depth, learn_rate, loss_reduction) %>%
  as.list

saveRDS(
  xgboost_parameters,
  file = sprintf(
    "./auxiliar/model_selection/hyperparameters/xgboost_%s.rds",
    outcome_column
  )
)

preds <- predict(final_xgboost_fit, new_data = df_train, type = "prob") %>%
  rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
  .$`1`

hist(preds)
```



Minutes to run: 12.624

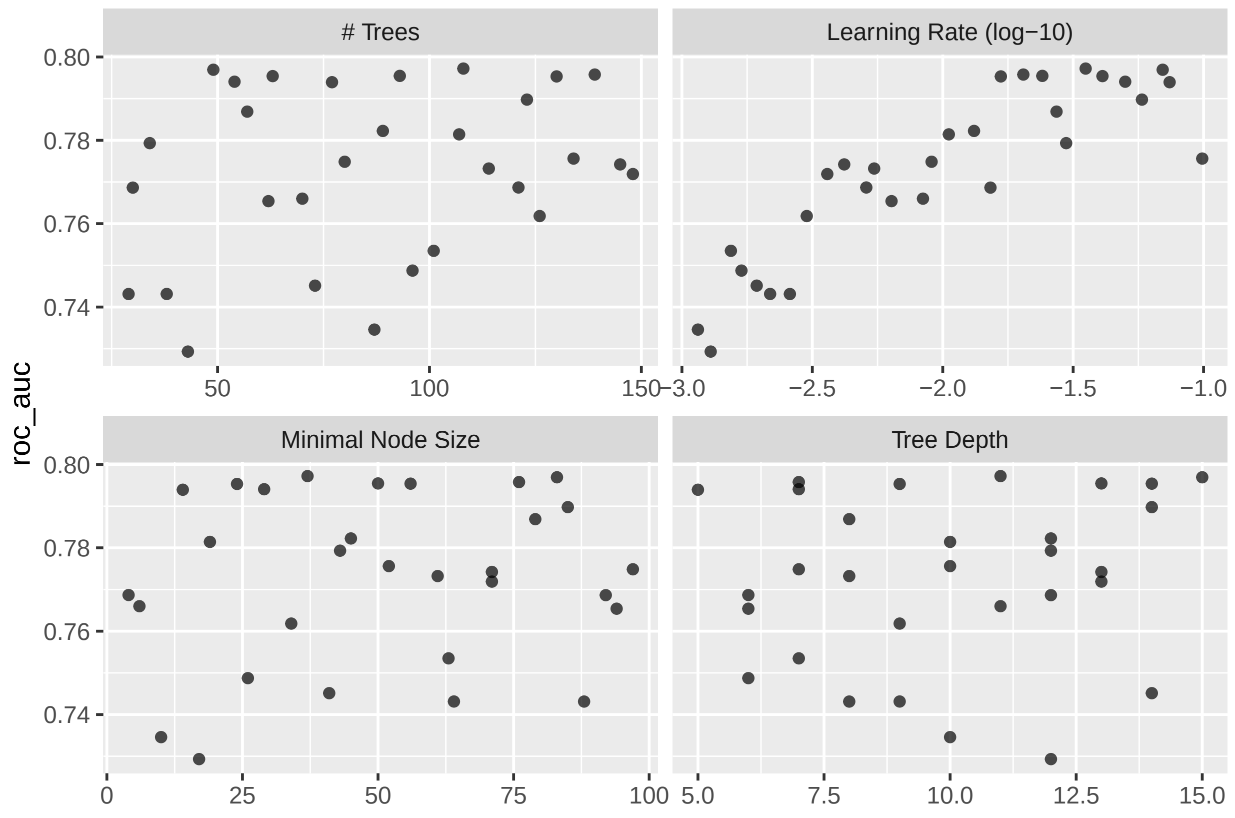
Boosted Tree (LightGBM)

```
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(5L, 15L)),
  learn_rate(range = c(-3, -1), trans = log10_trans()),
  size = grid_size
)

lightgbm_workflow <-
```

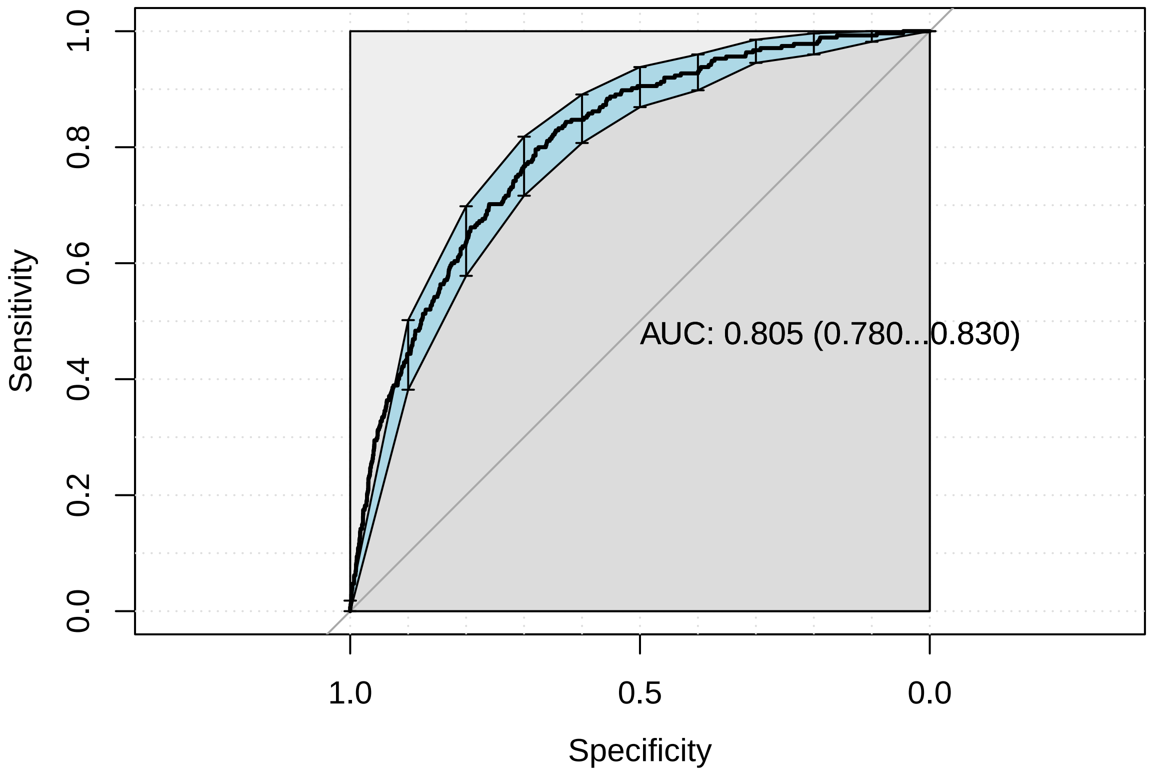
```
  workflow() %>%
  add_recipe(lightgbm_recipe) %>%
  add_model(lightgbm_spec)

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   108    37         11     0.0353 roc_auc binary     0.797    10  0.0100 Preprocessor1_Model11
## 2    49    83         15     0.0697 roc_auc binary     0.797    10  0.0110 Preprocessor1_Model25
## 3   139    76          7     0.0204 roc_auc binary     0.796    10  0.0100 Preprocessor1_Model23
## 4    93    50         13     0.0241 roc_auc binary     0.795    10  0.0102 Preprocessor1_Model15
## 5    63    56         14     0.0410 roc_auc binary     0.795    10  0.0101 Preprocessor1_Model17
```

```
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)
```



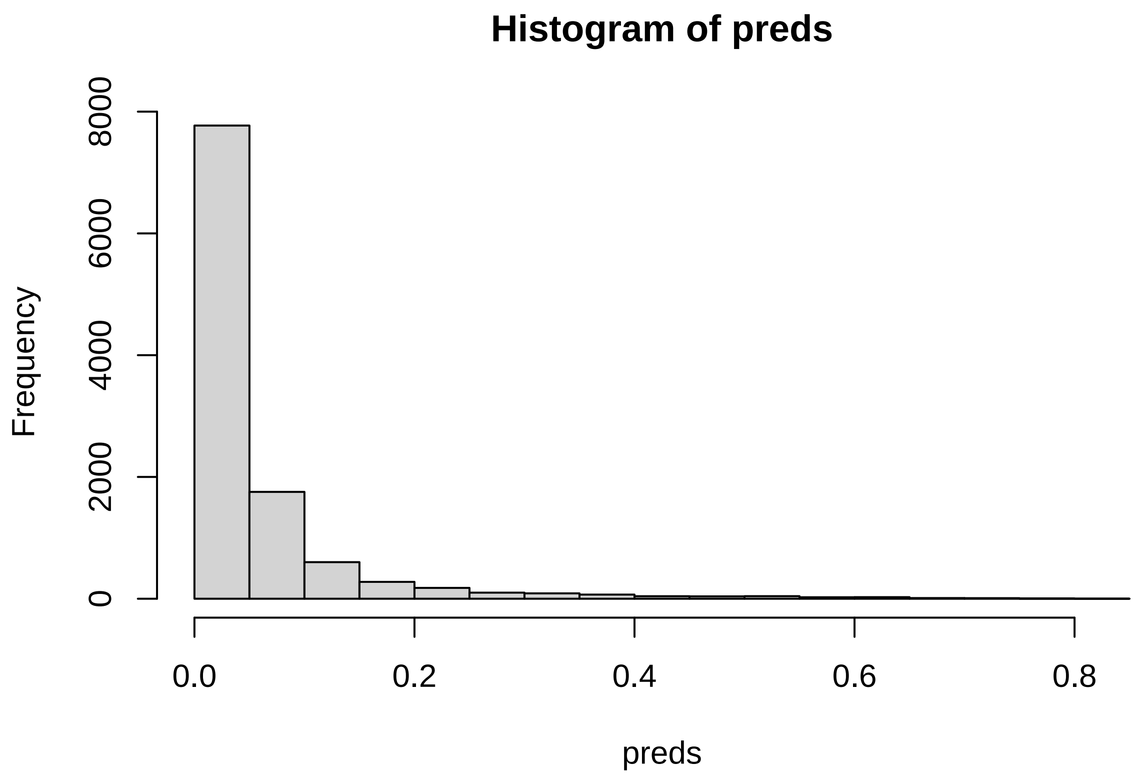
```
##   |
##
##        'Positive' Class : 0
##
```

```
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= 108
## .  min_n = integer 1= 37
## .  tree_depth = integer 1= 11
## .  learn_rate = double 1= 0.035312
```

```
saveRDS(
  lightgbm_parameters,
  file = sprintf(
    "./auxiliar/model_selection/hyperparameters/lightgbm_%s.rds",
    outcome_column
  )
)
```

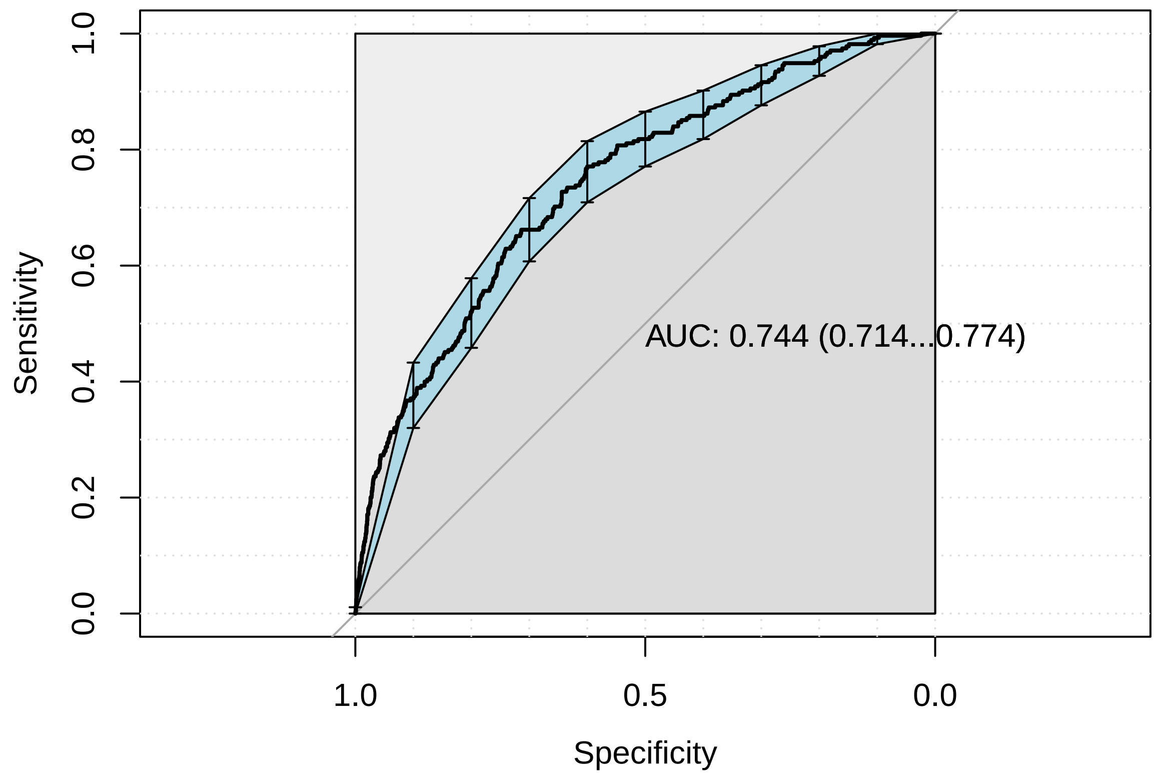
Minutes to run: 3.682



Minutes to run: 0.005

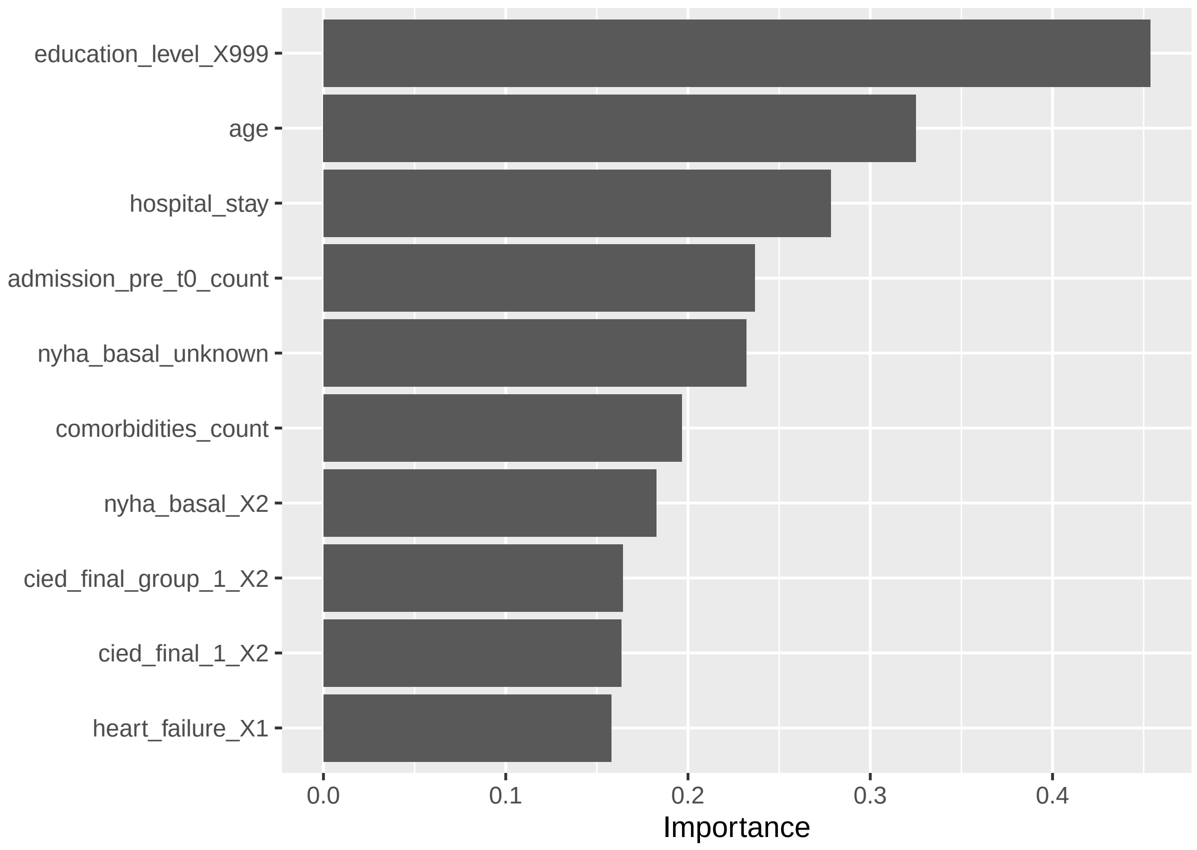
GLM

```
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())

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)
```

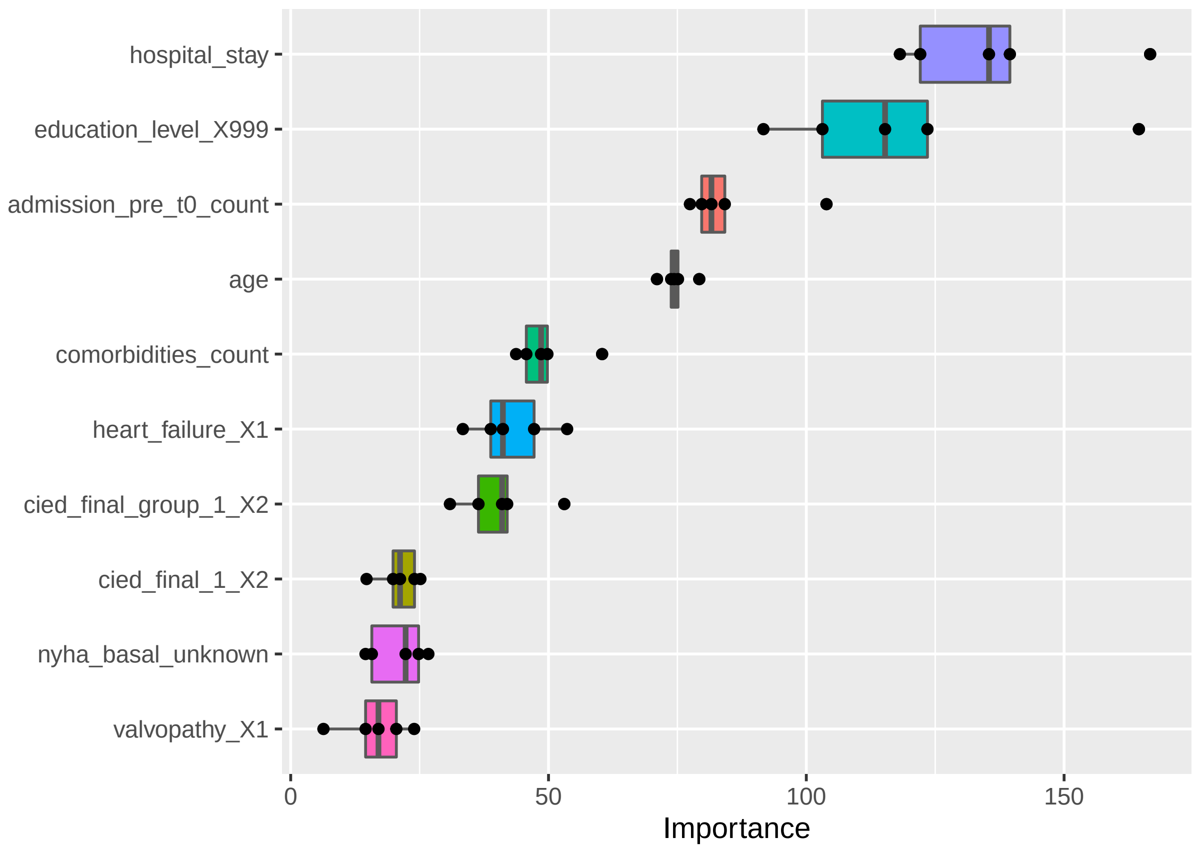


```
##   |
##
##                Accuracy : 0.7108
##                  95% CI : (0.6976, 0.7237)
##     No Information Rate : 0.9419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1245
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7138
##             Specificity : 0.6618
##          Pos Pred Value : 0.9716
##          Neg Pred Value : 0.1249
##              Prevalence : 0.9419
##          Detection Rate : 0.6723
##    Detection Prevalence : 0.6920
##       Balanced Accuracy : 0.6878
##
##        'Positive' Class : 0
##
```

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

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



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



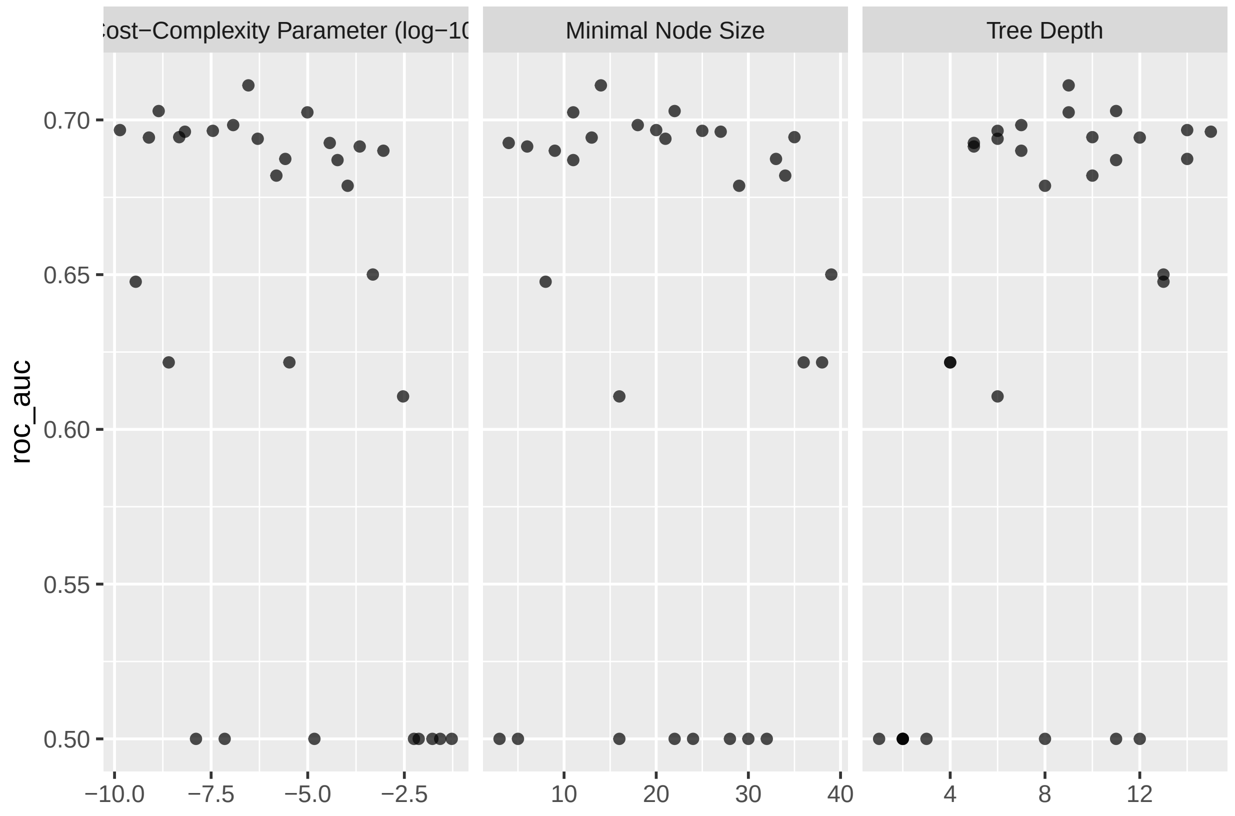
Minutes to run: 2.725

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: 60 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        2.21e- 4          5     6 accuracy binary     0.939    10 0.00229 Preprocessor1_Model01
##  2        2.21e- 4          5     6 roc_auc  binary     0.691    10 0.0148  Preprocessor1_Model01
##  3        2.67e- 2          8    32 accuracy binary     0.942    10 0.00259 Preprocessor1_Model02
##  4        2.67e- 2          8    32 roc_auc  binary     0.5      10 0       Preprocessor1_Model02
##  5        2.63e- 6         14    33 accuracy binary     0.936    10 0.00303 Preprocessor1_Model03
##  6        2.63e- 6         14    33 roc_auc  binary     0.687    10 0.0181  Preprocessor1_Model03
##  7        7.44e- 3          3    30 accuracy binary     0.942    10 0.00259 Preprocessor1_Model04
##  8        7.44e- 3          3    30 roc_auc  binary     0.5      10 0       Preprocessor1_Model04
##  9        7.76e-10         12    13 accuracy binary     0.926    10 0.00256 Preprocessor1_Model05
## 10        7.76e-10         12    13 roc_auc  binary     0.694    10 0.0128  Preprocessor1_Model05
## # ... with 50 more rows
```

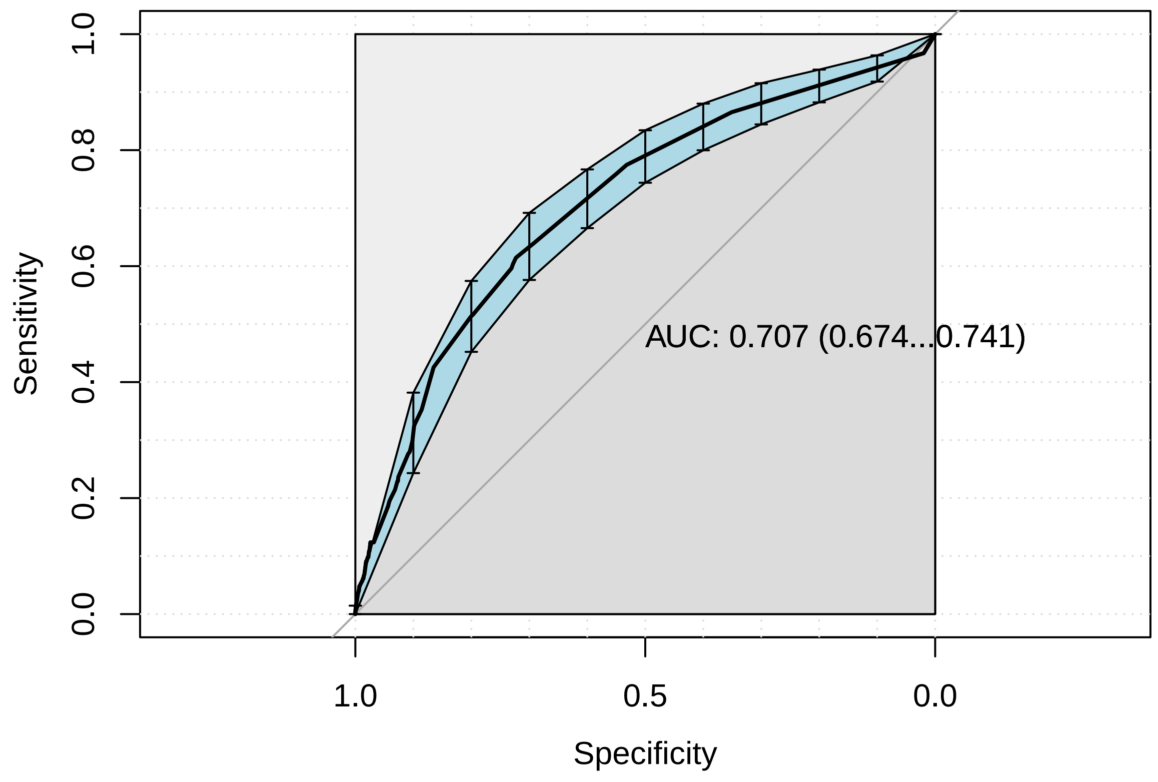
```
autoplot(tree_tune, metric = "roc_auc")
```



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

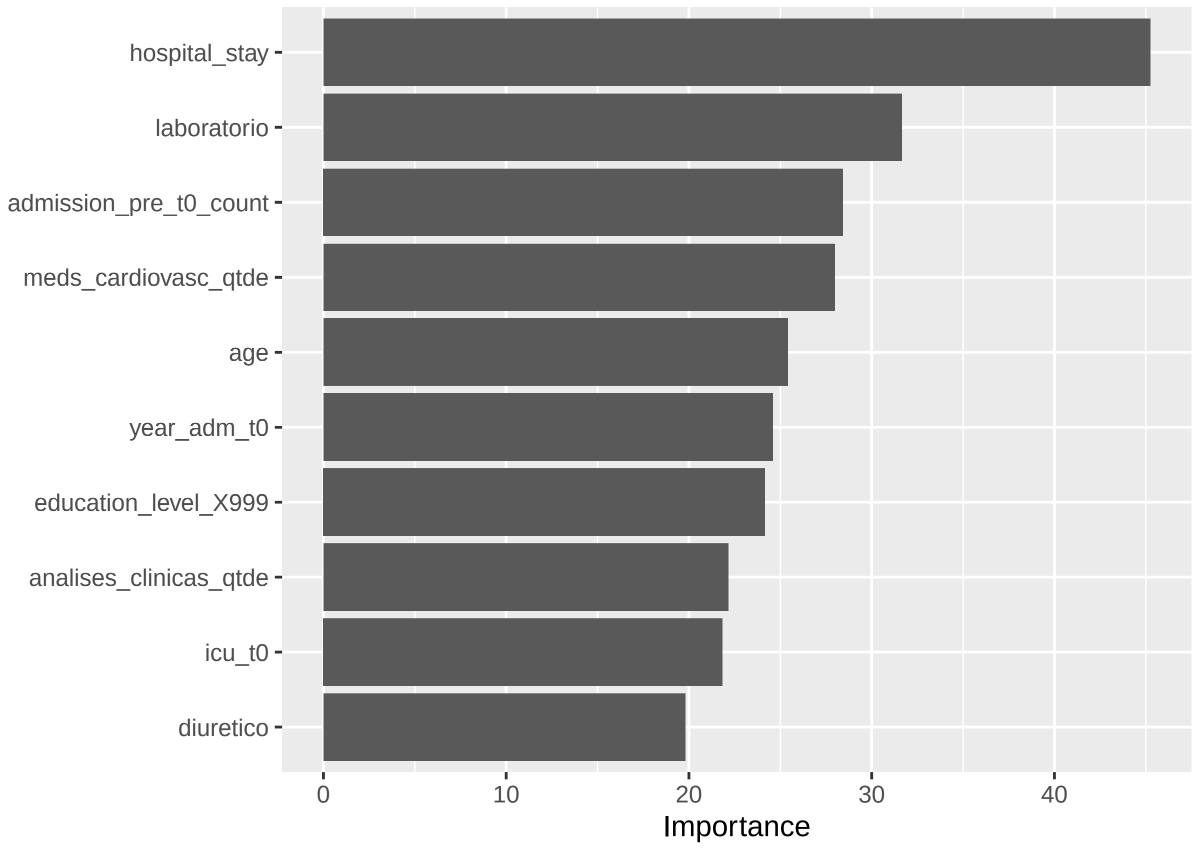
```
## # A tibble: 5 x 9
##   cost_complexity tree_depth min_n .metric .estimator  mean     n std_err .config
##             <dbl>      <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1        2.91e- 7          9    14 roc_auc binary     0.711    10  0.0135 Preprocessor1_Model25
## 2        1.39e- 9         11    22 roc_auc binary     0.703    10  0.0145 Preprocessor1_Model18
## 3        9.79e- 6          9    11 roc_auc binary     0.702    10  0.0136 Preprocessor1_Model17
## 4        1.18e- 7          7    18 roc_auc binary     0.698    10  0.0156 Preprocessor1_Model16
## 5        1.38e-10         14    20 roc_auc binary     0.697    10  0.0109 Preprocessor1_Model29
```

```
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)
```



```
##   |
##                   Kappa : 0.1154
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7230
##             Specificity : 0.6145
##          Pos Pred Value : 0.9681
##          Neg Pred Value : 0.1205
##              Prevalence : 0.9419
##          Detection Rate : 0.6810
##    Detection Prevalence : 0.7034
##       Balanced Accuracy : 0.6688
##
##        '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')
```

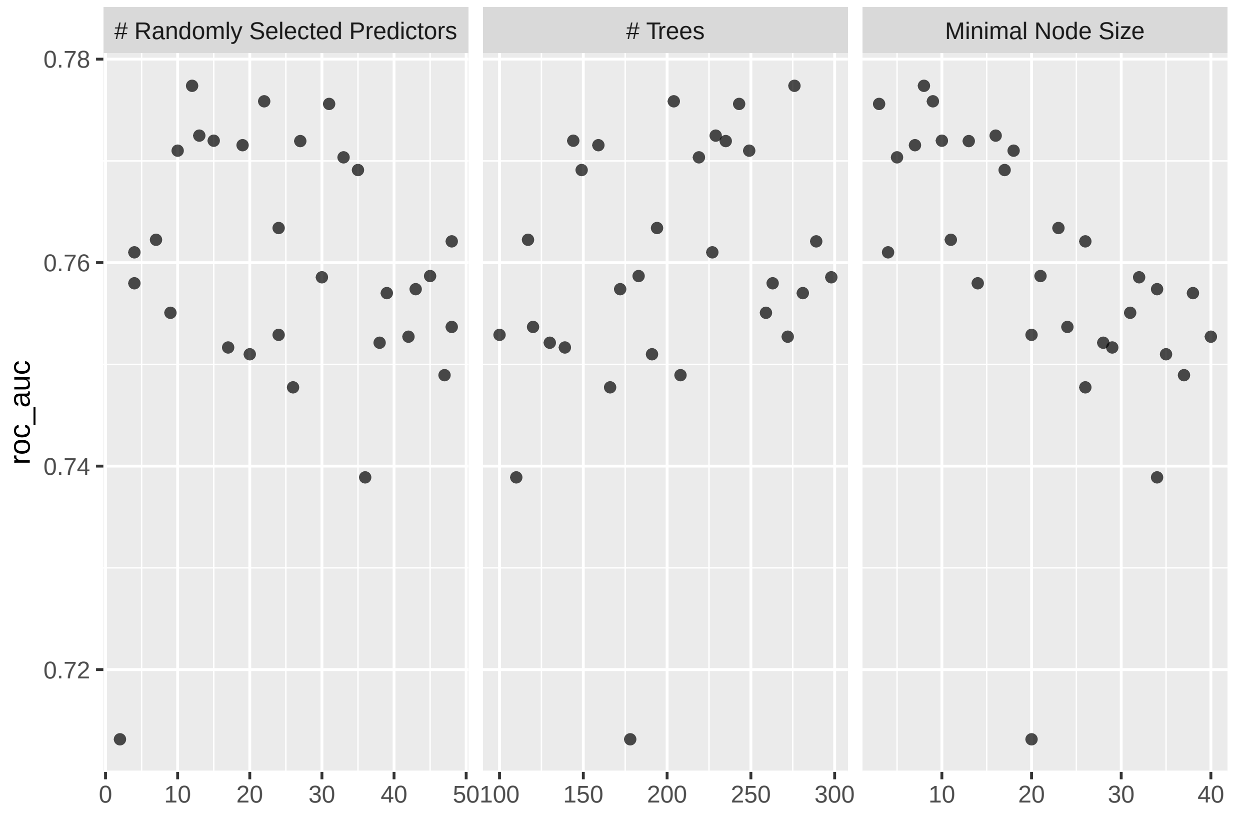
Minutes to run: 10.365

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: 60 x 9
##     mtry trees min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1    24   194    23 accuracy binary     0.942    10 0.00249 Preprocessor1_Model01
##  2    24   194    23 roc_auc  binary     0.763    10 0.0118  Preprocessor1_Model01
##  3    19   159     7 accuracy binary     0.942    10 0.00254 Preprocessor1_Model02
##  4    19   159     7 roc_auc  binary     0.772    10 0.0109  Preprocessor1_Model02
##  5    24   100    20 accuracy binary     0.942    10 0.00268 Preprocessor1_Model03
##  6    24   100    20 roc_auc  binary     0.753    10 0.0135  Preprocessor1_Model03
##  7    39   281    38 accuracy binary     0.942    10 0.00269 Preprocessor1_Model04
##  8    39   281    38 roc_auc  binary     0.757    10 0.00947 Preprocessor1_Model04
##  9    15   144    10 accuracy binary     0.942    10 0.00270 Preprocessor1_Model05
## 10    15   144    10 roc_auc  binary     0.772    10 0.0108  Preprocessor1_Model05
## # ... with 50 more rows
```

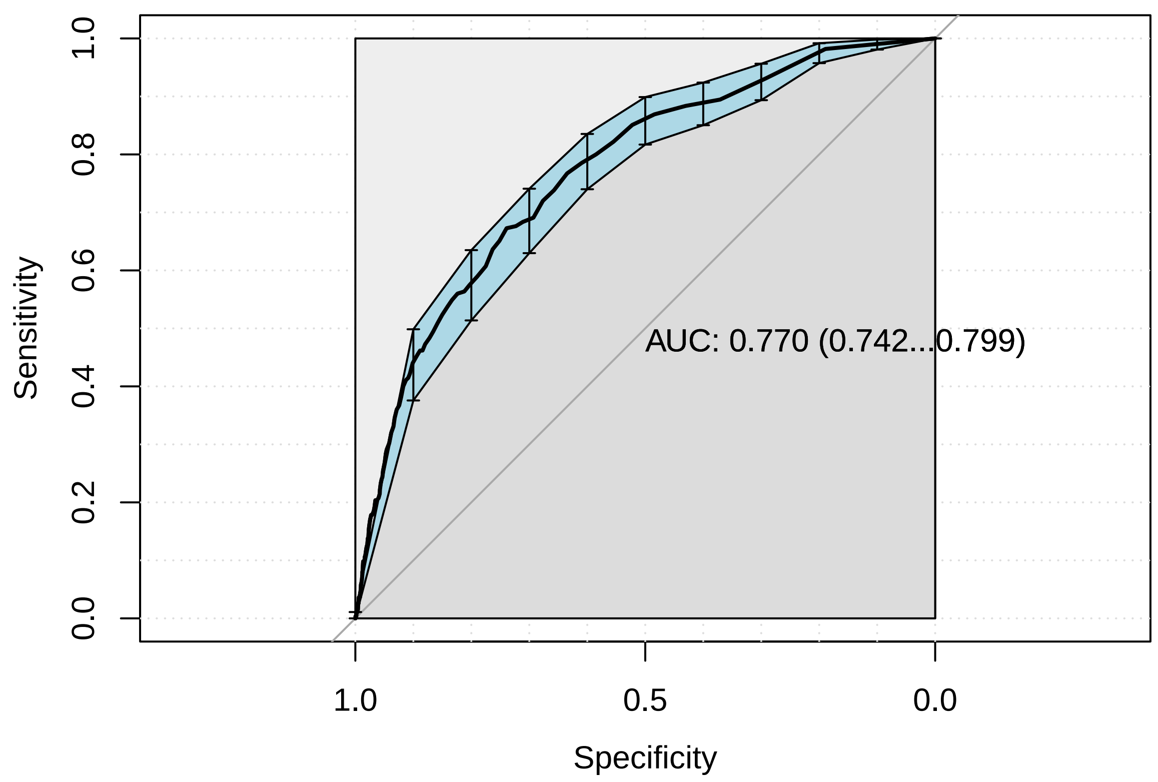
```r
autoplot(rf_tune, metric = "roc_auc")
```



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

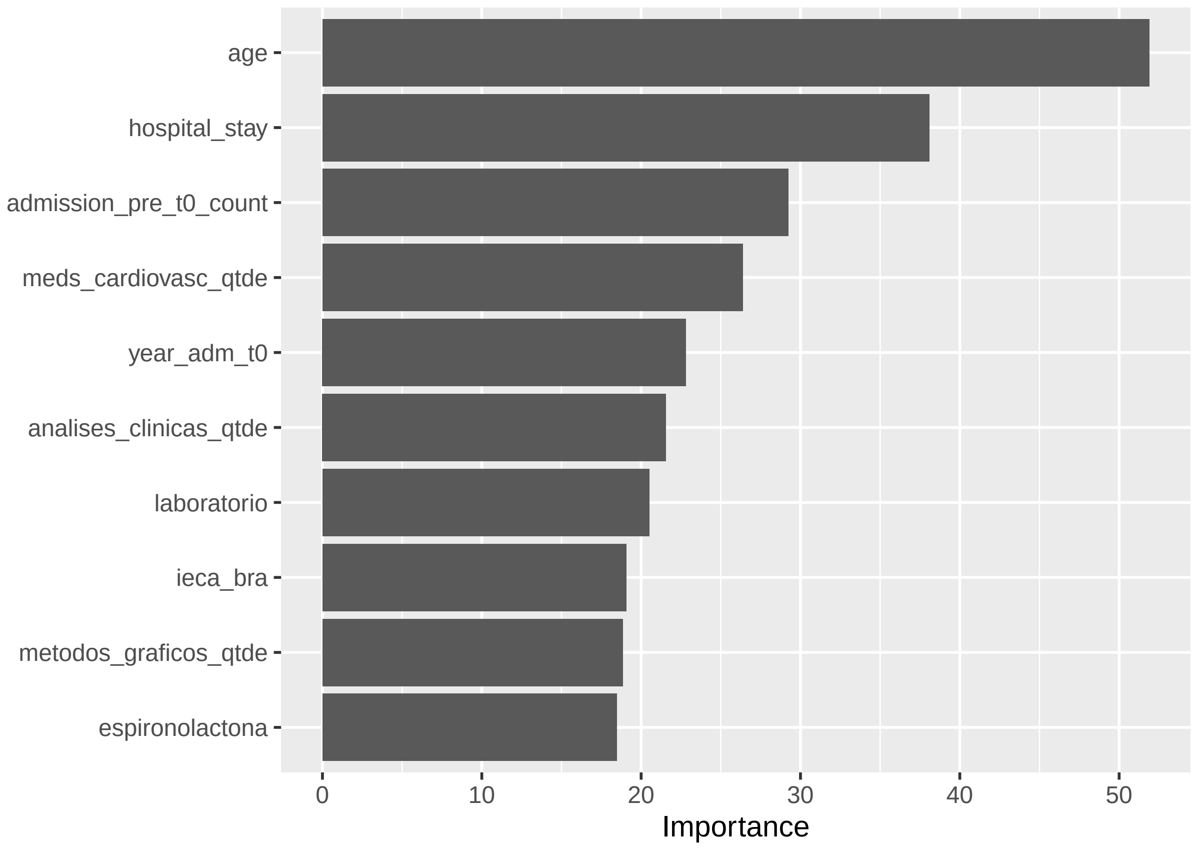
```
## # A tibble: 5 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    12   276     8 roc_auc binary     0.777    10 0.0108  Preprocessor1_Model16
## 2    22   204     9 roc_auc binary     0.776    10 0.00969 Preprocessor1_Model23
## 3    31   243     3 roc_auc binary     0.776    10 0.0114  Preprocessor1_Model20
## 4    13   229    16 roc_auc binary     0.772    10 0.0122  Preprocessor1_Model25
## 5    15   144    10 roc_auc binary     0.772    10 0.0108  Preprocessor1_Model05
```

```
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)
```



```
##   |
```

```
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')
```



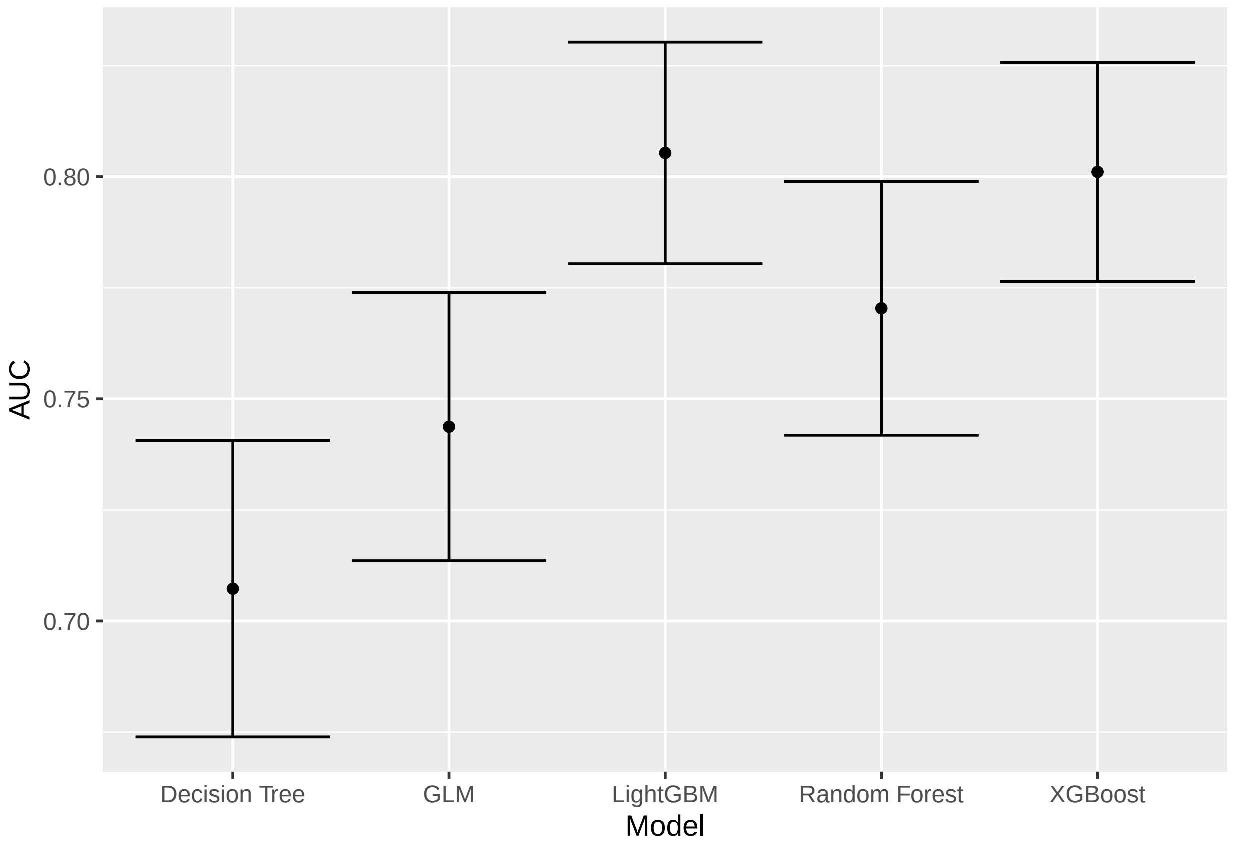
```
# extract_vip(final_rf_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 132.569

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()
```



```
saveRDS(df_auc, sprintf("./auxiliar/model_selection/performance/%s.RData", outcome_column))
```

Minutes to run: 0.002