

Model Selection - readmission_180d

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 (968 bytes)
## . outcome_column = character 1= readmission_180d
## . 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: 0.005

```
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 = 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

```

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. patient_state
## 05. underlying_heart_disease
## 06. heart_disease
## 07. nyha_basal
## 08. prior_mi
## 09. heart_failure
## 10. af
## 11. cardiac_arrest
## 12. transplant
## 13. valvopathy

```

14. endocardites
15. diabetes
16. renal_failure
17. hemodialysis
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. icu_t0
28. dialysis_t0
29. n_procedure_t0
30. admission_t0_emergency
31. aco
32. antiarritmico
33. betabloqueador
34. ieca_bra
35. dva
36. digoxina
37. estatina
38. diuretico
39. vasodilatador
40. insuf_cardiaca
41. espirolactona
42. bloq_calcio
43. antiplaquetario_ev
44. insulina
45. anticonvulsivante
46. psicofarmacos
47. antifungico
48. antiviral
49. antiretroviral
50. classe_meds_qtde
51. meds_cardiovasc_qtde
52. meds_antimicrobianos
53. vni
54. ventilacao_mecanica
55. cec
56. transplante_cardiaco
57. cir_toracica
58. outros_proced_cirurgicos
59. icp
60. intervencao_cv
61. angioplastia
62. cateterismo
63. eletrofisiologia
64. cateter_venoso_central
65. proced_invasivos_qtde
66. cve_desf
67. transfusao
68. interconsulta
69. equipe_multiprof
70. holter
71. teste_esforco
72. espiro_ergoespiro
73. tilt_teste
74. metodos_graficos_qtde

```
## 75. laboratorio
## 76. cultura
## 77. analises_clinicas_qtde
## 78. citologia
## 79. biopsia
## 80. histopatologia_qtde
## 81. angio_rm
## 82. angio_tc
## 83. arteriografia
## 84. cintilografia
## 85. ecocardiograma
## 86. endoscopia
## 87. flebografia
## 88. pet_ct
## 89. ultrassom
## 90. tomografia
## 91. ressonancia
## 92. exames_imagem_qtde
## 93. bic
## 94. mpp
## 95. 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)
```

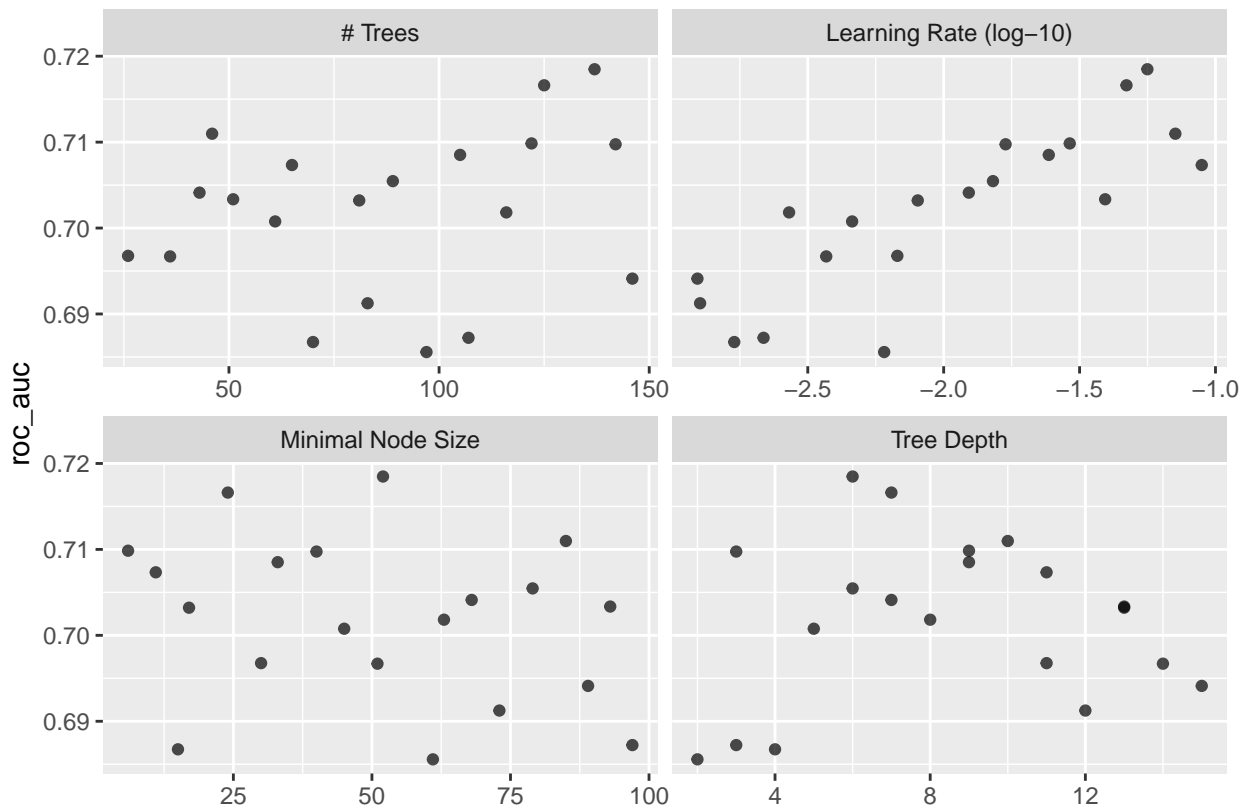
```
xgboost_tune <-
  xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)
```

```
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   137    52         6    0.0562 roc_auc binary    0.718    10  0.0120 Preprocessor1_Model11
## 2   125    24         7    0.0471 roc_auc binary    0.717    10  0.0114 Preprocessor1_Model05
## 3    46    85        10    0.0712 roc_auc binary    0.711    10  0.0101 Preprocessor1_Model17
## 4   122     6         9    0.0291 roc_auc binary    0.710    10  0.0138 Preprocessor1_Model01
## 5   142    40         3    0.0169 roc_auc binary    0.710    10  0.0104 Preprocessor1_Model08
```

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

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

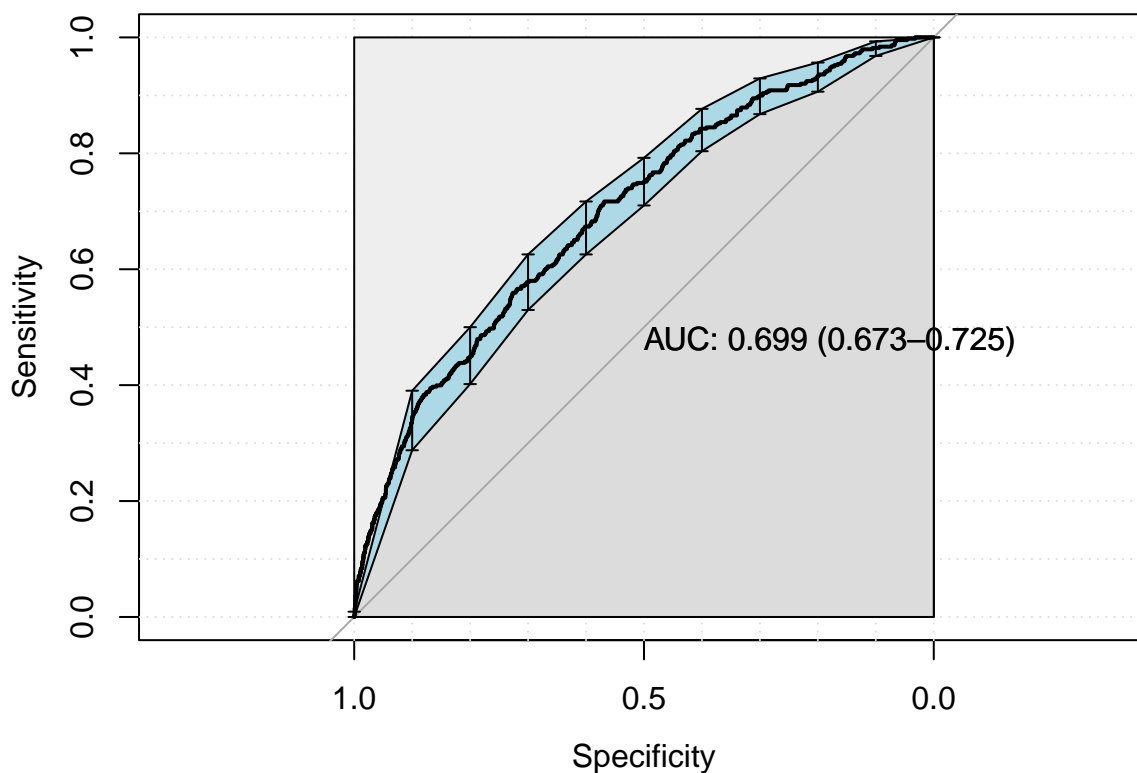


```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

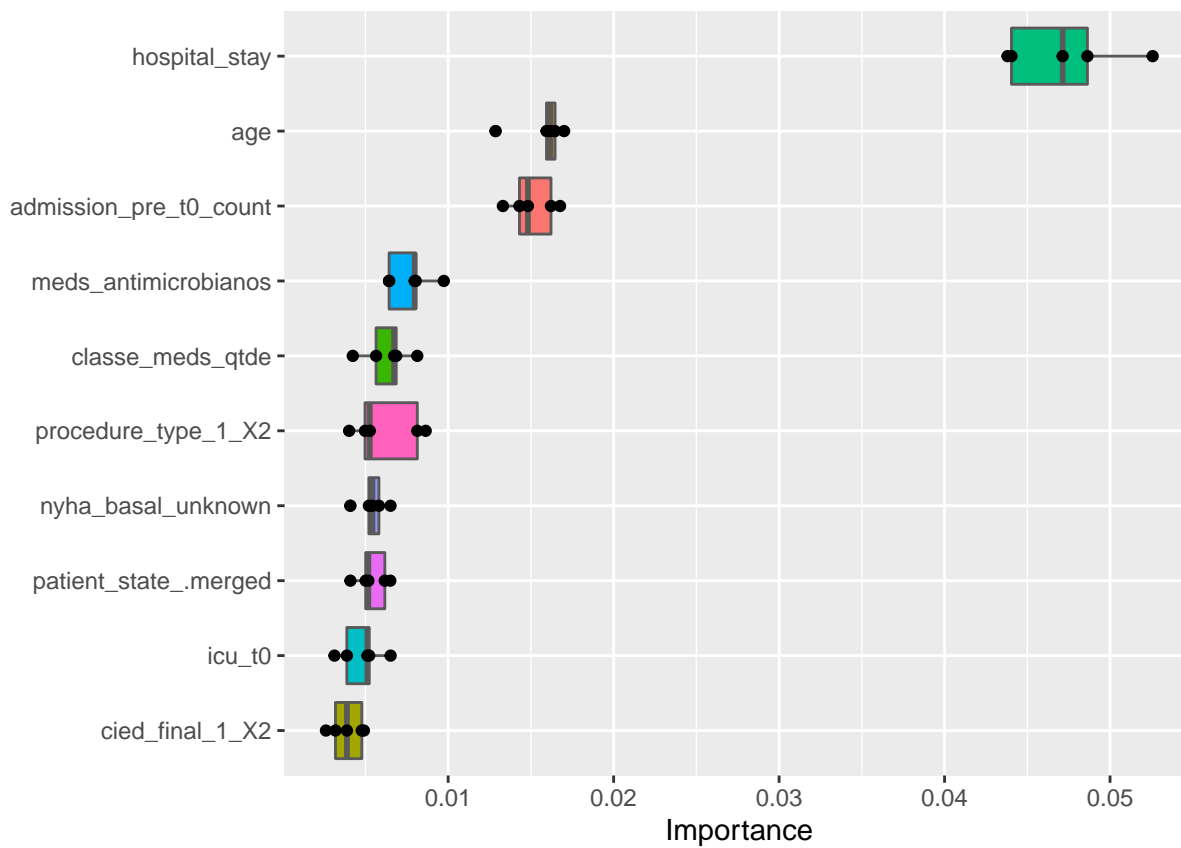
last_xgboost_fit <-
  final_xgboost_workflow %>%
  last_fit(df_split)

final_xgboost_fit <- extract_workflow(last_xgboost_fit)

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



```
## [1] "Optimal Threshold: 0.10"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 3118 193
##    1 1174 245
##
##              Accuracy : 0.711
##              95% CI : (0.6978, 0.7239)
##    No Information Rate : 0.9074
##    P-Value [Acc > NIR] : 1
##
##              Kappa : 0.1425
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.7265
##              Specificity : 0.5594
##              Pos Pred Value : 0.9417
##              Neg Pred Value : 0.1727
##              Prevalence : 0.9074
##              Detection Rate : 0.6592
##              Detection Prevalence : 0.7000
##              Balanced Accuracy : 0.6429
##
##              'Positive' Class : 0
##
extract_vip(final_xgboost_fit, pred_wrapper = predict,
            reference_class = "0")
```



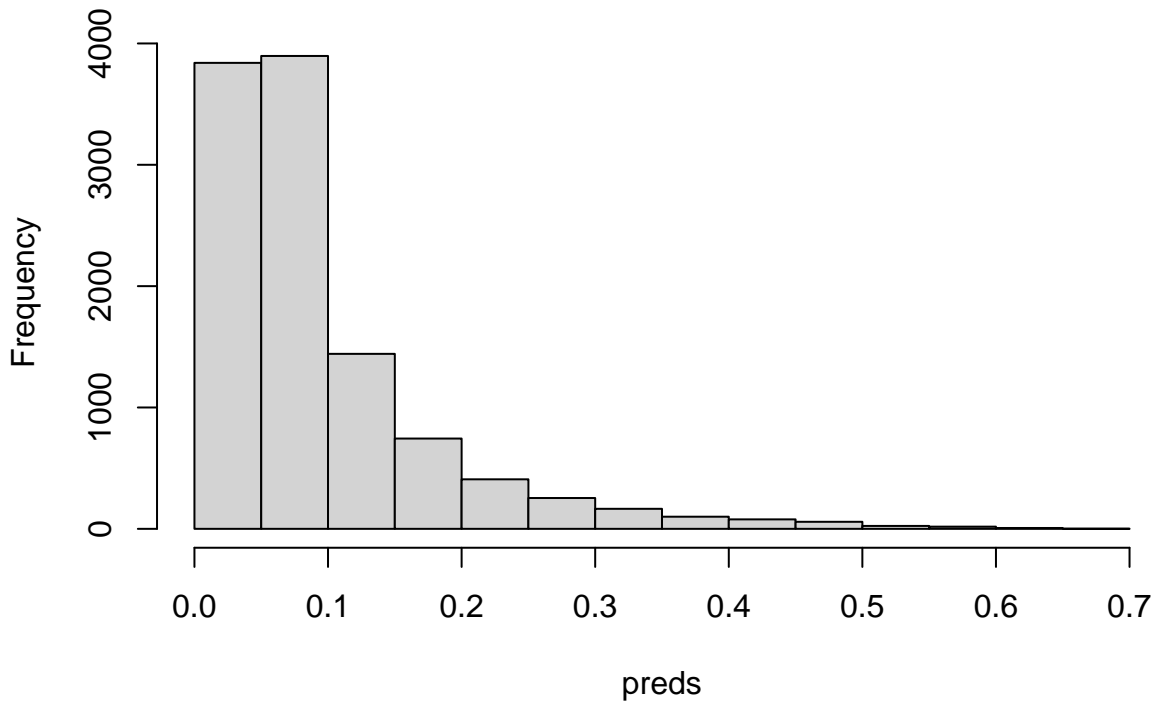
```
xgboost_parameters <- xgboost_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(-.metric, -.estimator, -.config, -mean, -n, -std_err) %>%
  as.list

# 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)
```


Histogram of preds



Minutes to run:

5.351

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)

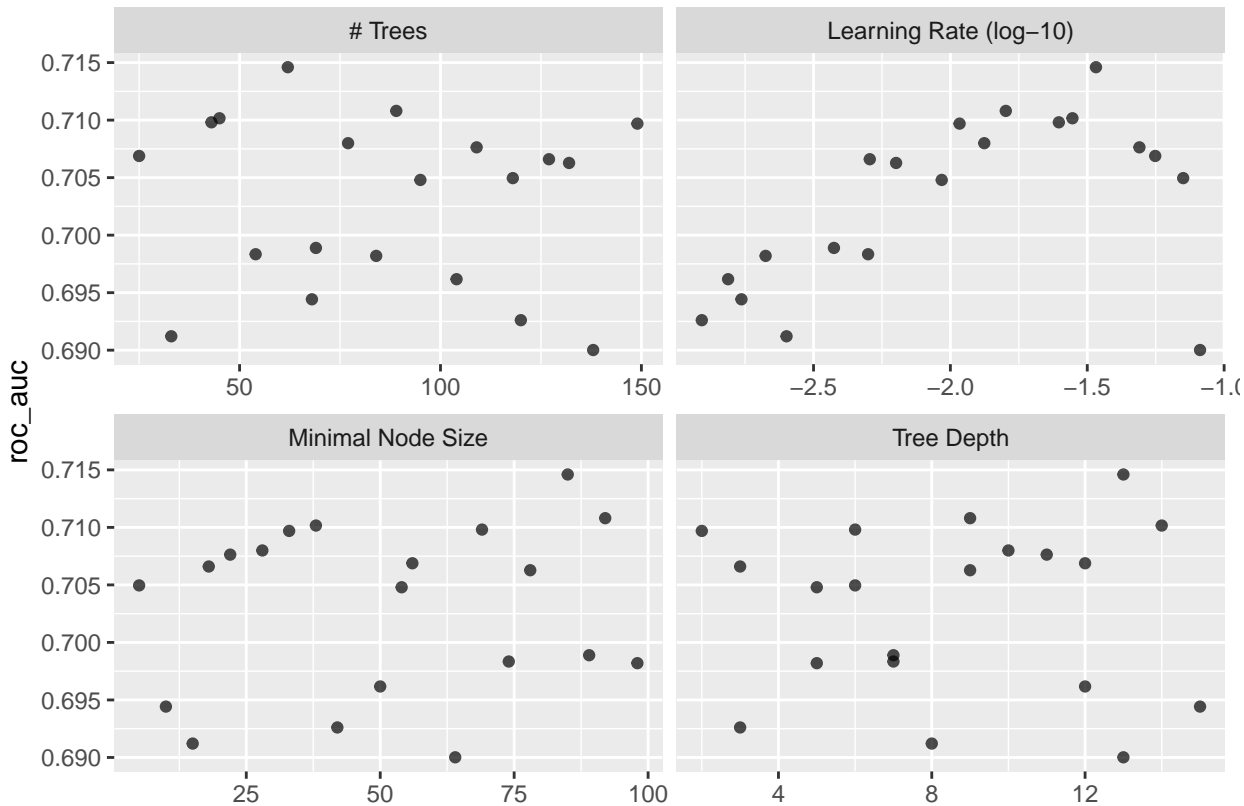
lightgbm_tune %>%
  show_best("roc_auc")

## # 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     62     85        13    0.0340 roc_auc binary  0.715    10  0.0116 Preprocessor1_Model17
## 2     89     92         9    0.0159 roc_auc binary  0.711    10  0.0119 Preprocessor1_Model19
## 3     45     38        14    0.0279 roc_auc binary  0.710    10  0.0117 Preprocessor1_Model08
## 4     43     69         6    0.0249 roc_auc binary  0.710    10  0.0114 Preprocessor1_Model14
## 5    149     33         2    0.0108 roc_auc binary  0.710    10  0.0102 Preprocessor1_Model07

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

autoplot(lightgbm_tune, metric = "roc_auc")

```



```

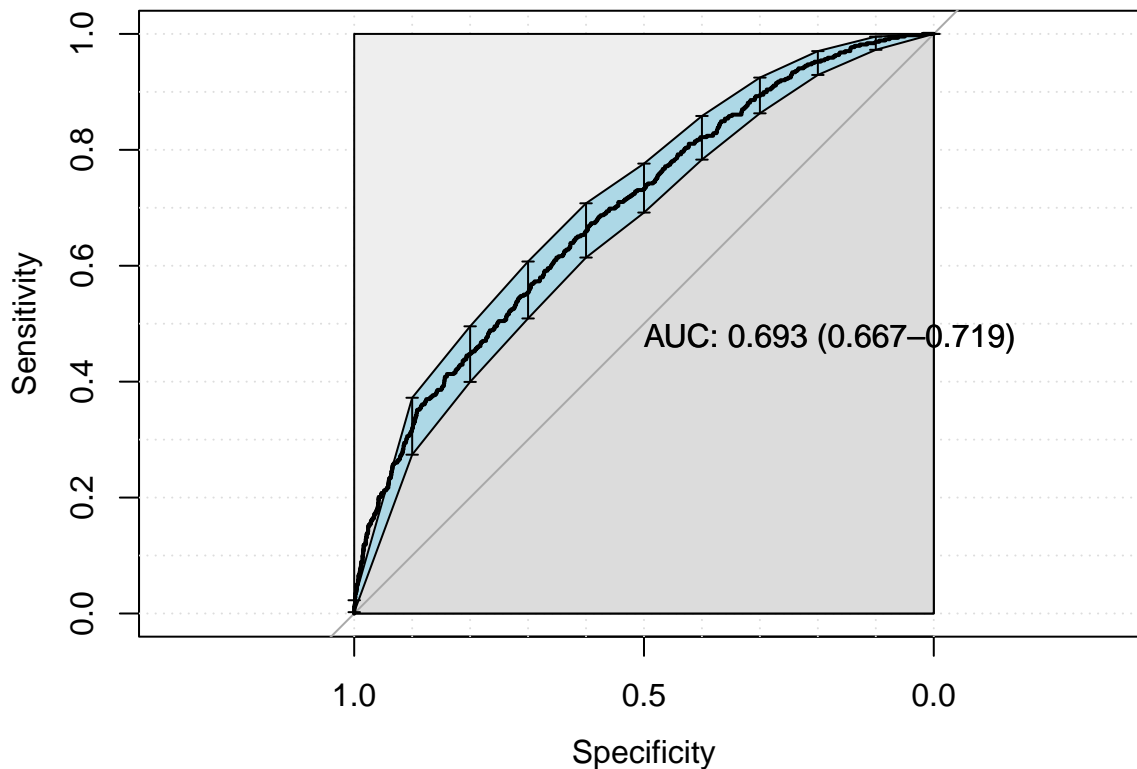
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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: 0.08"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 2651 154
##    1 1641 284
##
##              Accuracy : 0.6205
##              95% CI   : (0.6065, 0.6344)
##    No Information Rate : 0.9074
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1054
##
## Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6177
##              Specificity : 0.6484
##    Pos Pred Value   : 0.9451
##    Neg Pred Value   : 0.1475
##    Prevalence       : 0.9074
##    Detection Rate   : 0.5605
##    Detection Prevalence : 0.5930
##    Balanced Accuracy : 0.6330
##
##      '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)

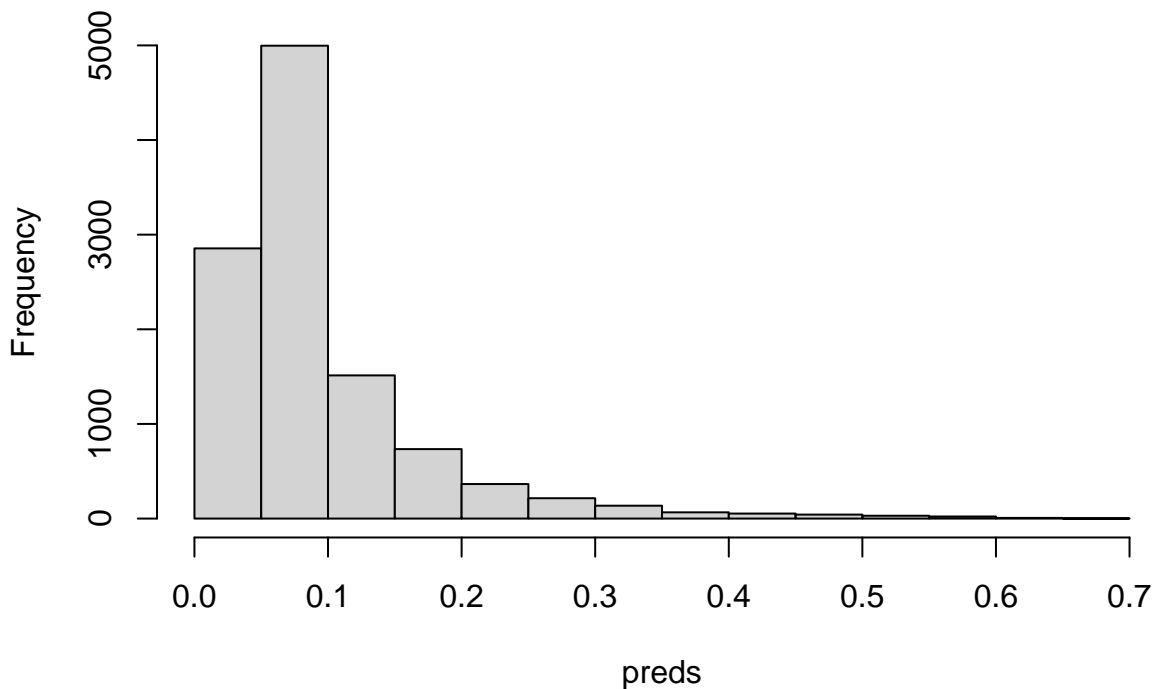
##  lightgbm_parameters = list 4 (736 bytes)
## .  trees = integer 1= 62
## .  min_n = integer 1= 85
## .  tree_depth = integer 1= 13
## .  learn_rate = double 1= 0.03403

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

```

Minutes to run: 3.683

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 <-

```

```

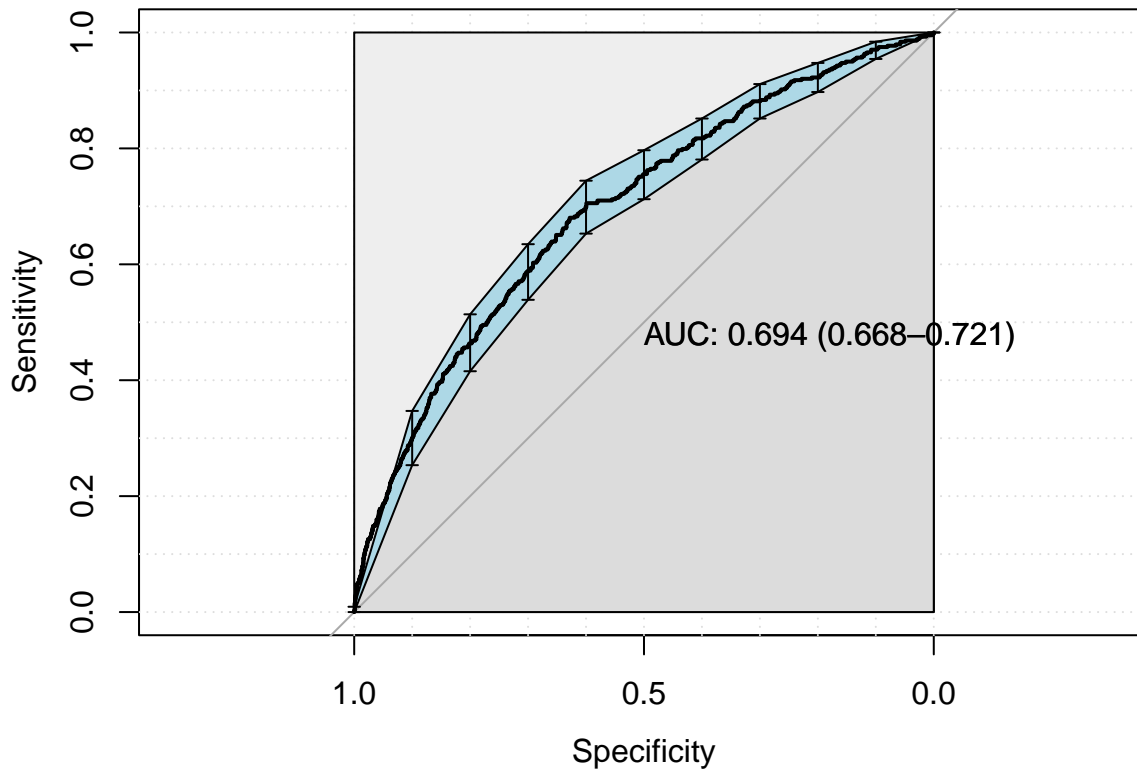
logistic_reg(penalty = 0) %>%
set_mode("classification") %>%
set_engine("glmnet")

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

glm_fit <- glmnet_workflow %>%
  fit(df_train)

glmnet_auc <- validation(glm_fit, df_test)

```



```

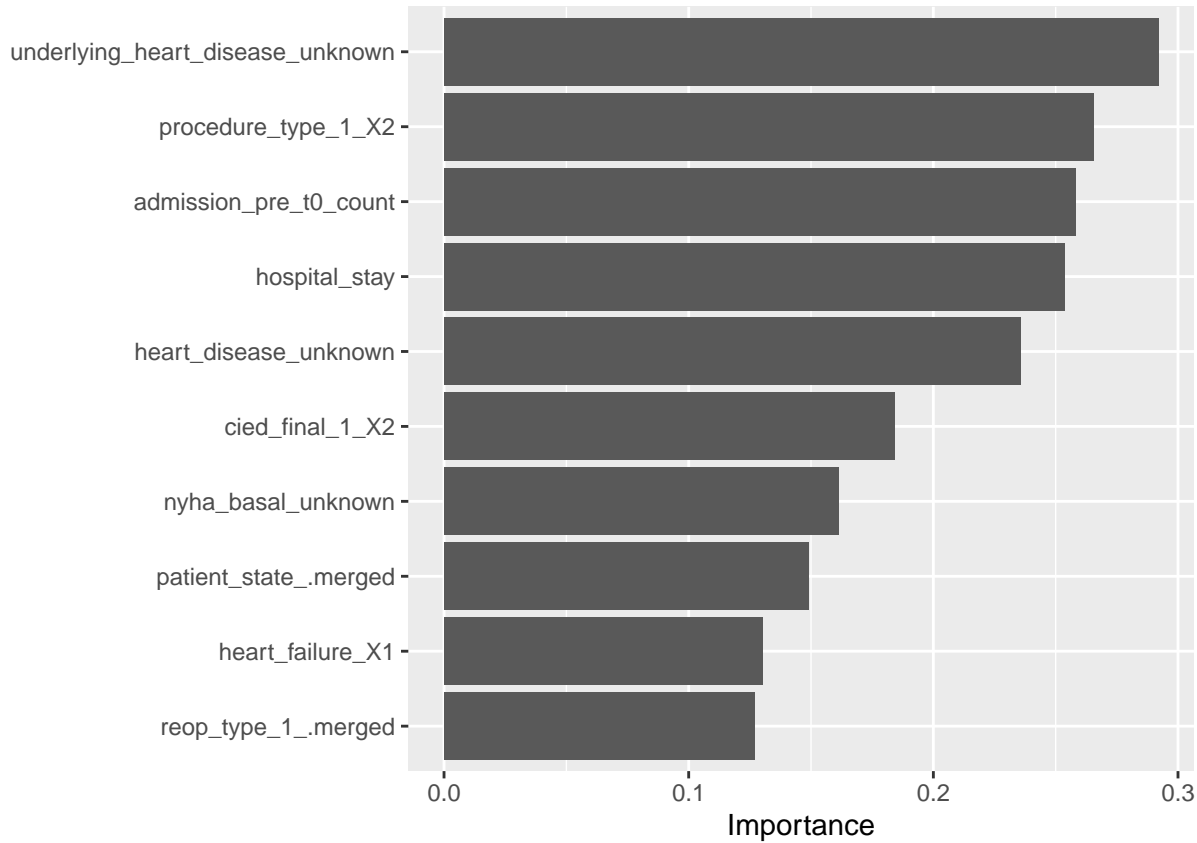
## [1] "Optimal Threshold: 0.08"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 2696  140
## 1 1596  298
##
##              Accuracy : 0.633
##              95% CI   : (0.6191, 0.6467)
##    No Information Rate : 0.9074
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1238
##
##  Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6281
##              Specificity : 0.6804

```

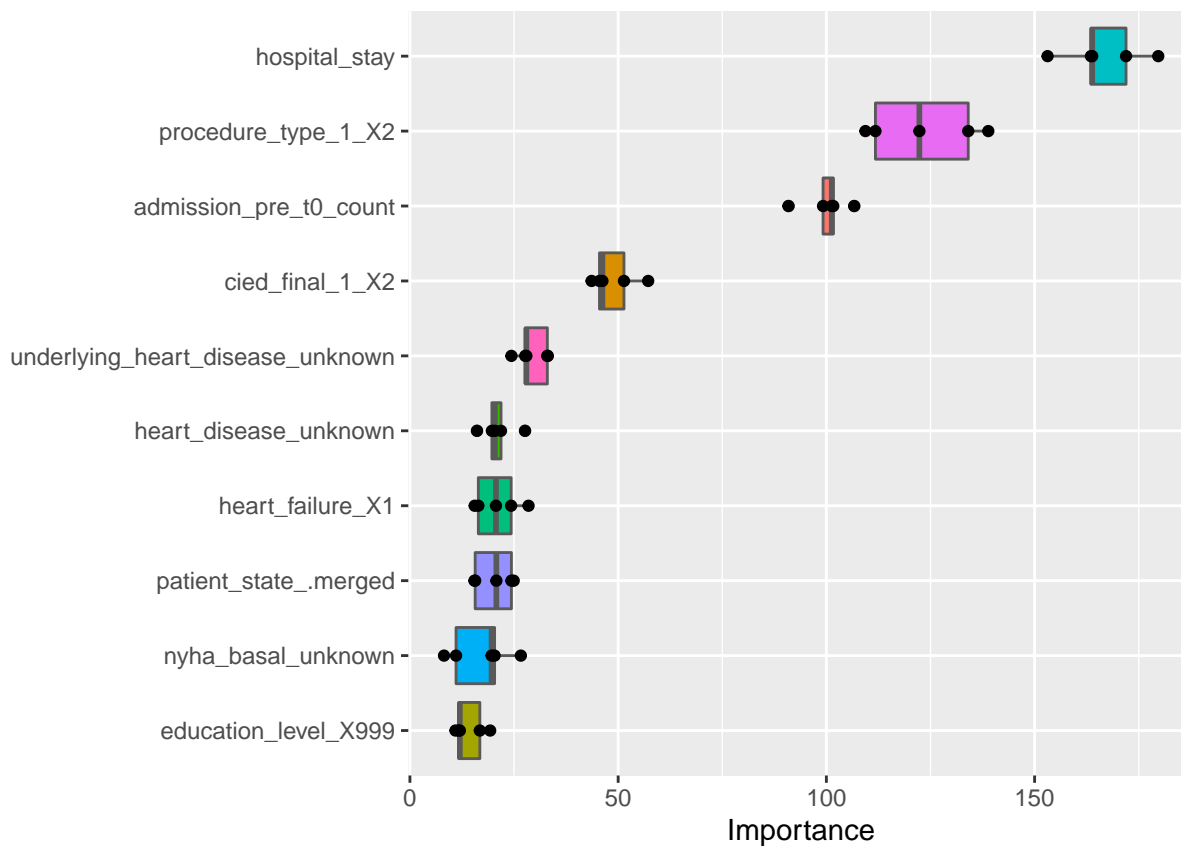
```
##          Pos Pred Value : 0.9506
##          Neg Pred Value : 0.1573
##          Prevalence : 0.9074
##          Detection Rate : 0.5700
##          Detection Prevalence : 0.5996
##          Balanced Accuracy : 0.6543
##
##          'Positive' Class : 0
##
```

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

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



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



Minutes to run:

2.041

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)

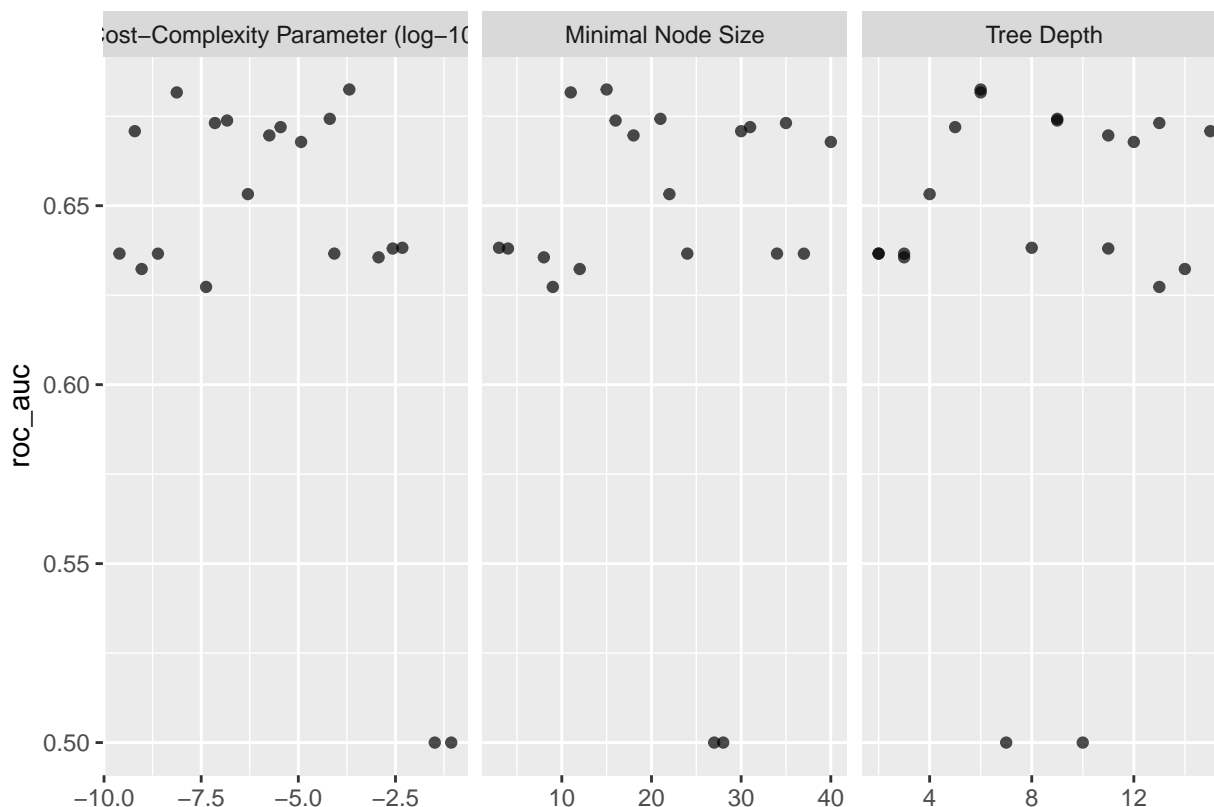
tree_workflow <-
  workflow() %>%
  add_recipe(tree_recipe) %>%
  add_model(tree_spec)

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      3.50e- 6         5     31 accuracy binary   0.904    10 0.00288 Preprocessor1_Model101
## 2      3.50e- 6         5     31 roc_auc  binary   0.672    10 0.0123  Preprocessor1_Model101
## 3      1.79e- 6        11     18 accuracy binary   0.893    10 0.00326 Preprocessor1_Model102
## 4      1.79e- 6        11     18 roc_auc  binary   0.670    10 0.0113  Preprocessor1_Model102
## 5      2.69e- 3        11      4 accuracy binary   0.903    10 0.00307 Preprocessor1_Model103
## 6      2.69e- 3        11      4 roc_auc  binary   0.638    10 0.00802 Preprocessor1_Model103
## 7      1.16e- 3         3      8 accuracy binary   0.905    10 0.00285 Preprocessor1_Model104
## 8      1.16e- 3         3      8 roc_auc  binary   0.636    10 0.00817 Preprocessor1_Model104
## 9      2.50e-10         2     24 accuracy binary   0.906    10 0.00278 Preprocessor1_Model105
## 10     2.50e-10         2     24 roc_auc  binary   0.637    10 0.00798 Preprocessor1_Model105
## # ... with 30 more rows
## # i Use 'print(n = ...)' to see more rows
```

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



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

```
## # A tibble: 5 x 9
##   cost_complexity tree_depth min_n .metric .estimator mean      n std_err .config
##   <dbl>          <int> <int> <chr>   <chr>    <dbl> <int>  <dbl> <chr>
## 1  0.000205         6     15 roc_auc  binary   0.682    10 0.00944 Preprocessor1_Model106
## 2  0.00000000745     6     11 roc_auc  binary   0.682    10 0.00919 Preprocessor1_Model117
## 3  0.0000645        9     21 roc_auc  binary   0.674    10 0.00824 Preprocessor1_Model108
## 4  0.000000147        9     16 roc_auc  binary   0.674    10 0.0103  Preprocessor1_Model116
## 5  0.0000000714     13     35 roc_auc  binary   0.673    10 0.0110  Preprocessor1_Model119
```



```

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

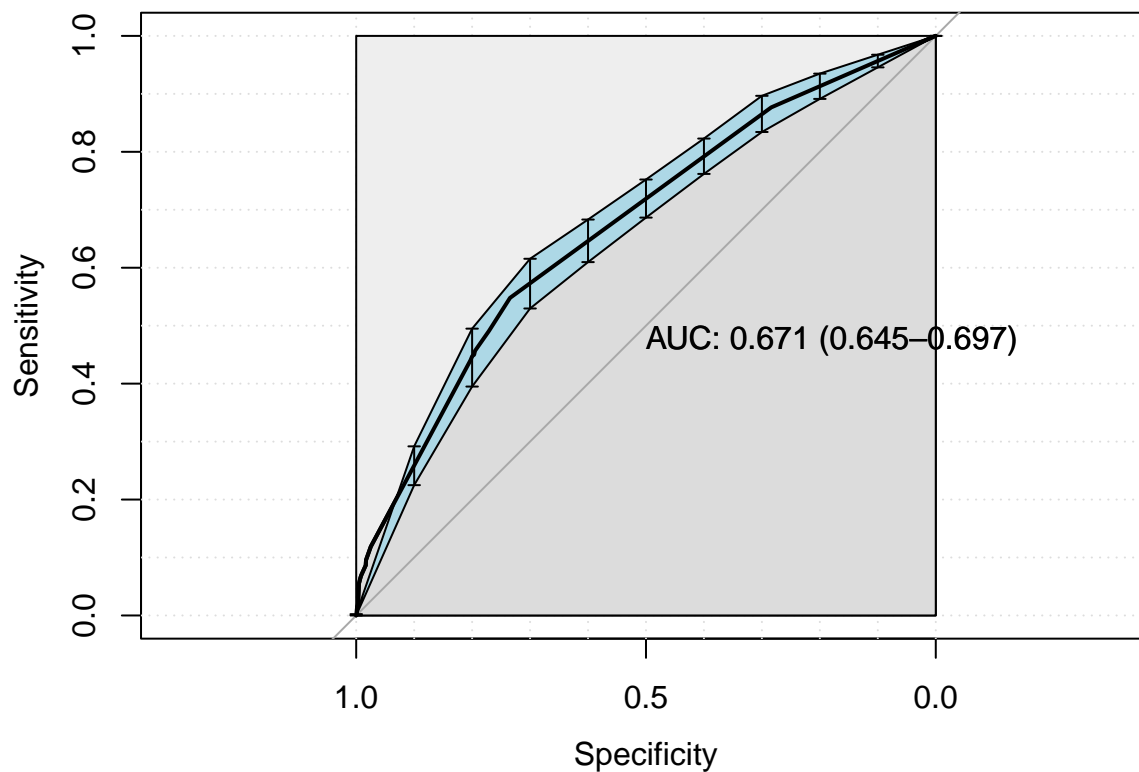
final_tree_workflow <-
  tree_workflow %>%
  finalize_workflow(best_tree)

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)

```



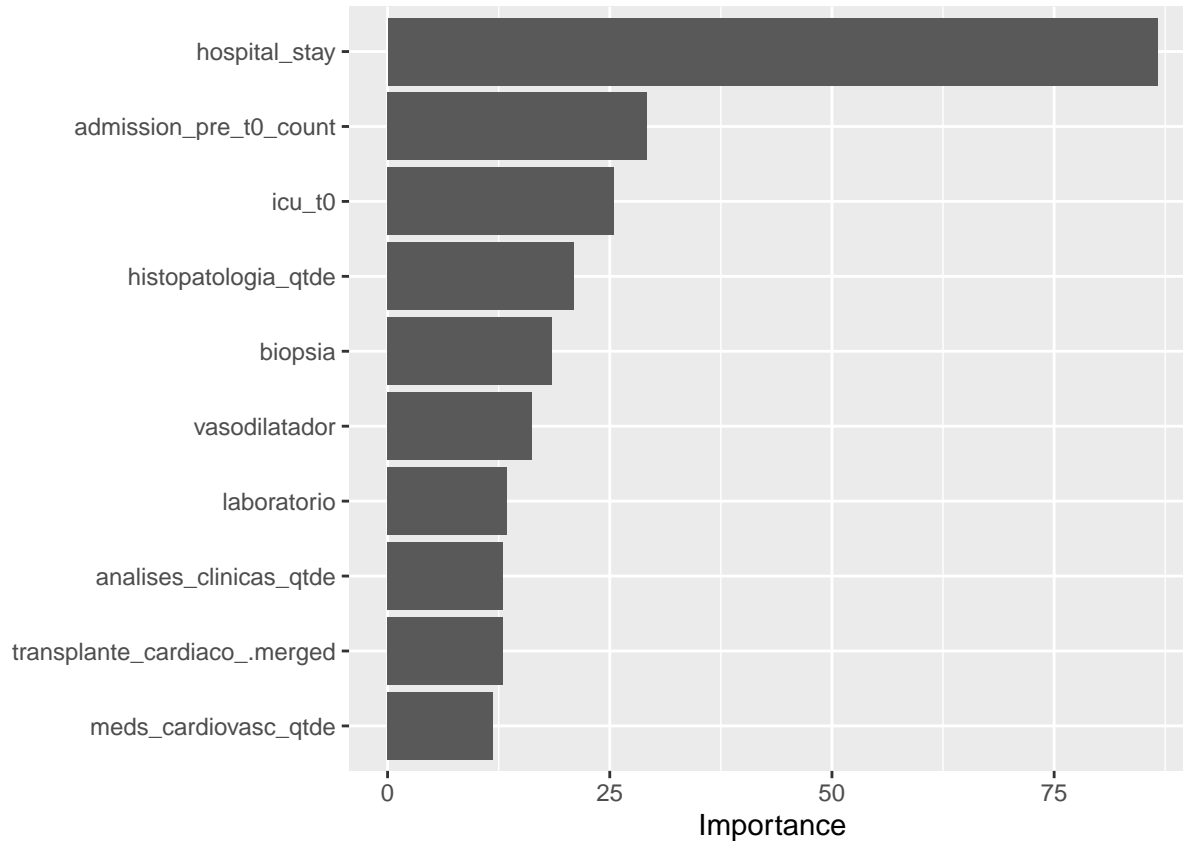
```

## [1] "Optimal Threshold: 0.09"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
## 0 3153  198
## 1 1139  240
##
##              Accuracy : 0.7173
##              95% CI : (0.7043, 0.7301)
##      No Information Rate : 0.9074
##      P-Value [Acc > NIR] : 1
##
##              Kappa : 0.1438
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.7346

```

```
##          Specificity : 0.5479
##          Pos Pred Value : 0.9409
##          Neg Pred Value : 0.1740
##          Prevalence : 0.9074
##          Detection Rate : 0.6666
##          Detection Prevalence : 0.7085
##          Balanced Accuracy : 0.6413
##
##          '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: 5.319

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)

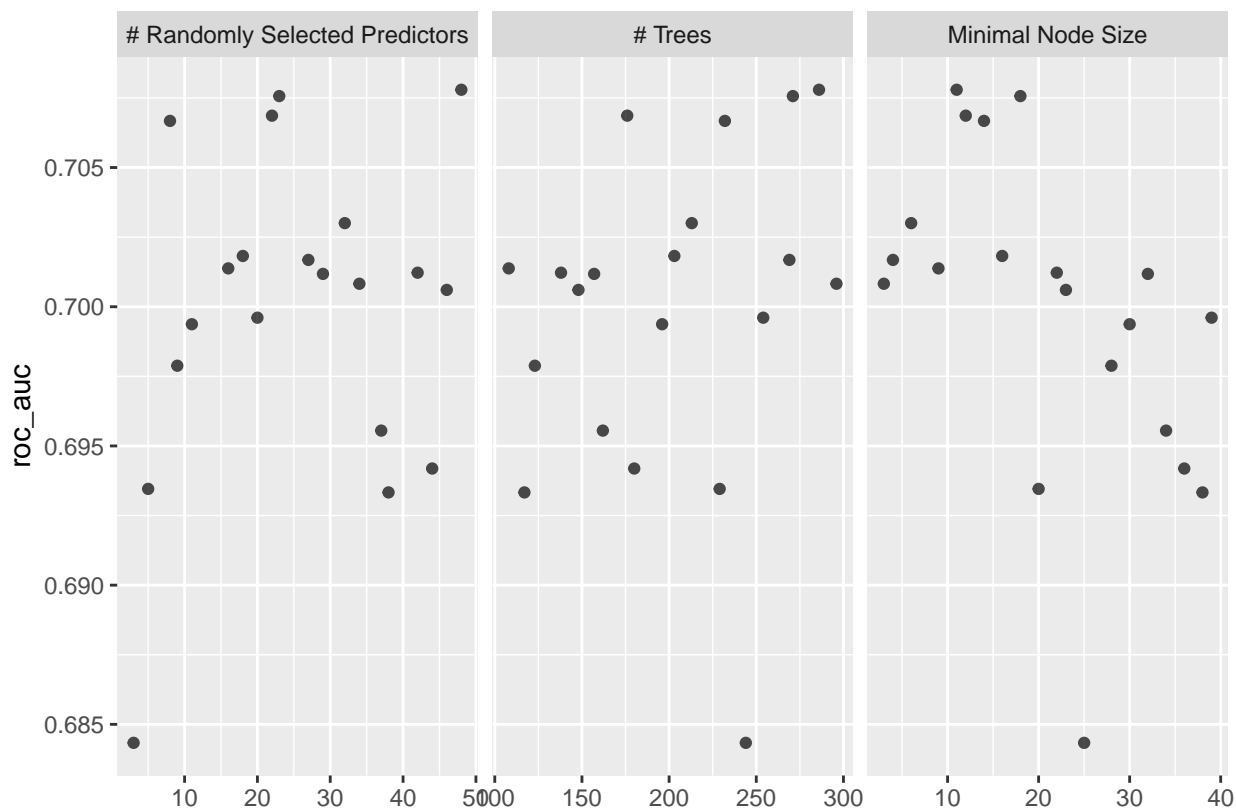
rf_tune <-
  rf_workflow %>%
  tune_grid(resamples = df_folds,
            grid = rf_grid)

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     34   296     3 accuracy binary    0.906    10 0.00274 Preprocessor1_Model01
## 2     34   296     3 roc_auc  binary    0.701    10 0.0140  Preprocessor1_Model01
## 3      3   244    25 accuracy binary    0.906    10 0.00281 Preprocessor1_Model02
## 4      3   244    25 roc_auc  binary    0.684    10 0.0113  Preprocessor1_Model02
## 5     22   176    12 accuracy binary    0.906    10 0.00264 Preprocessor1_Model03
## 6     22   176    12 roc_auc  binary    0.707    10 0.0136  Preprocessor1_Model03
## 7      8   232    14 accuracy binary    0.906    10 0.00273 Preprocessor1_Model04
## 8      8   232    14 roc_auc  binary    0.707    10 0.0126  Preprocessor1_Model04
## 9     29   157    32 accuracy binary    0.906    10 0.00264 Preprocessor1_Model05
## 10    29   157    32 roc_auc  binary    0.701    10 0.0129  Preprocessor1_Model05
## # ... with 30 more rows
## # i Use 'print(n = ...)' to see more rows

autoplot(rf_tune, metric = "roc_auc")

```



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

```
## # A tibble: 5 x 9
##   mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>    <dbl> <int>   <dbl> <chr>
## 1    48   286    11 roc_auc binary   0.708    10  0.0125 Preprocessor1_Model19
## 2    23   271    18 roc_auc binary   0.708    10  0.0131 Preprocessor1_Model12
## 3    22   176    12 roc_auc binary   0.707    10  0.0136 Preprocessor1_Model03
## 4     8   232    14 roc_auc binary   0.707    10  0.0126 Preprocessor1_Model04
## 5    32   213     6 roc_auc binary   0.703    10  0.0142 Preprocessor1_Model16
```

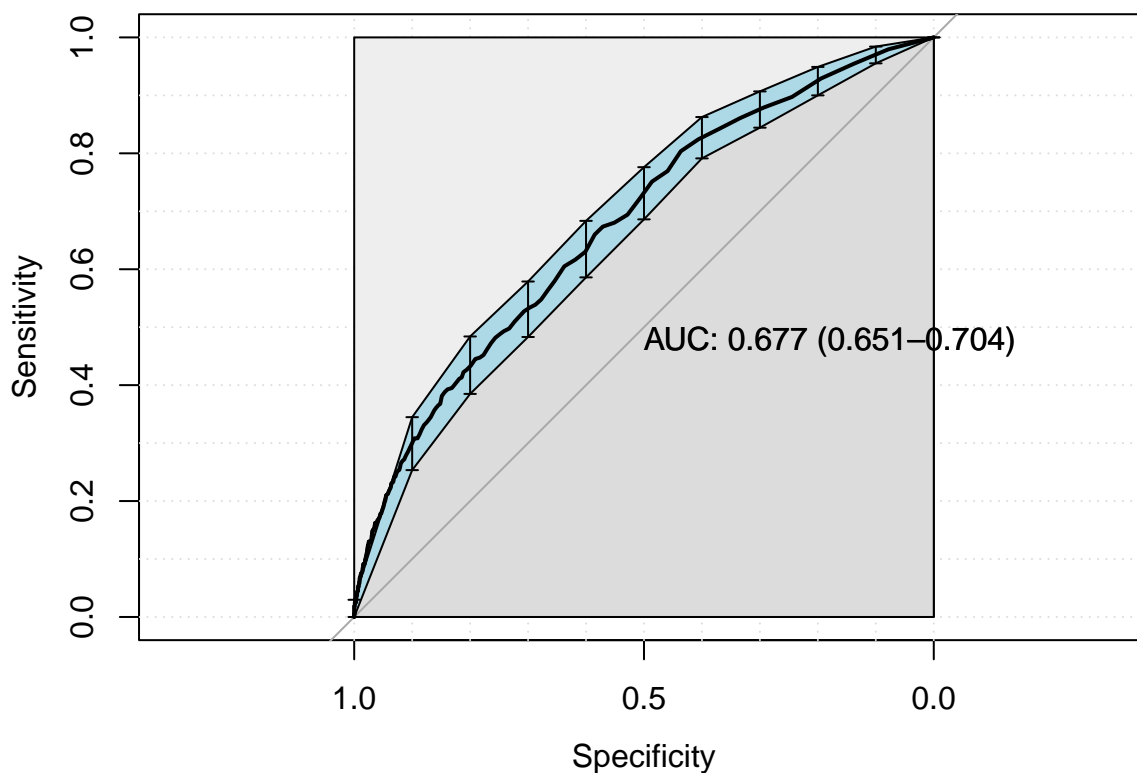
```
best_rf <- rf_tune %>%
  select_best("roc_auc")
```

```
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)
```

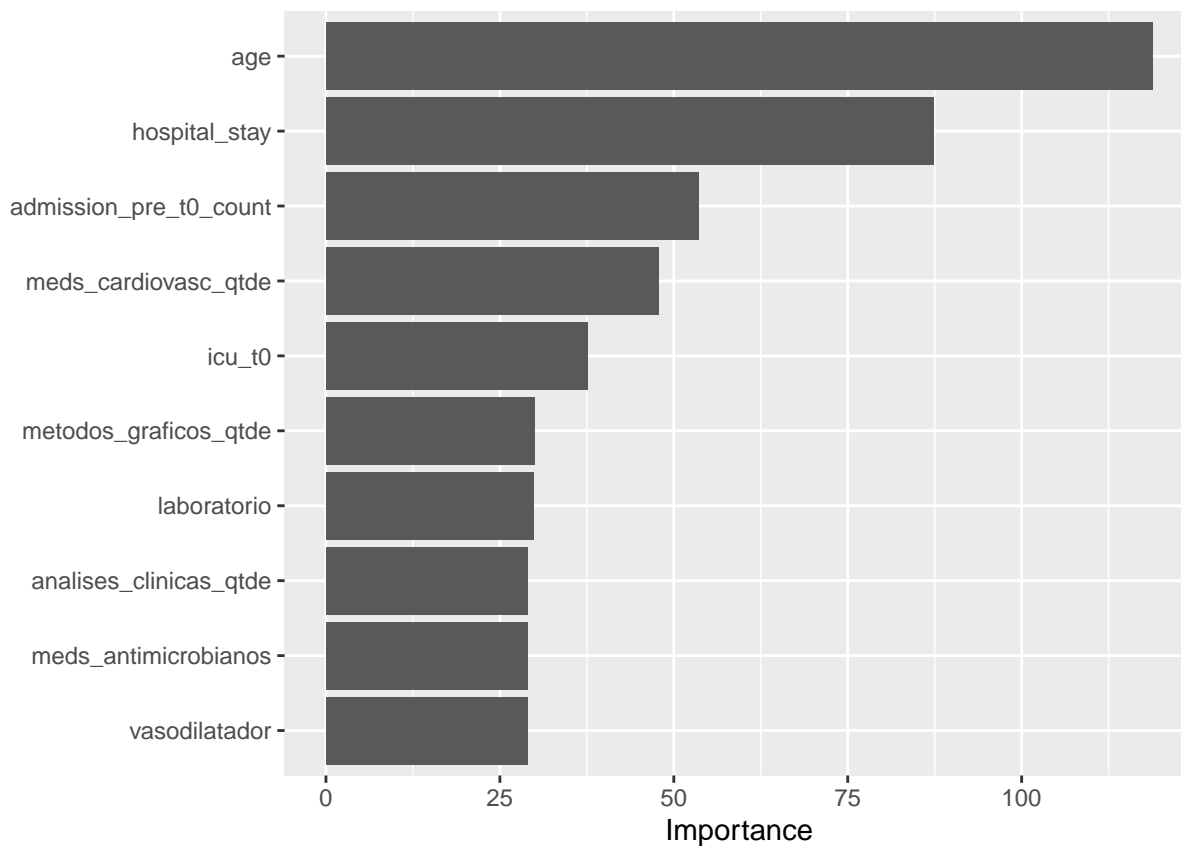
```
rf_auc <- validation(final_rf_fit, df_test)
```



```
## [1] "Optimal Threshold: 0.05"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 2511  149
##    1 1781  289
##
##              Accuracy : 0.592
##              95% CI   : (0.5778, 0.606)
##    No Information Rate : 0.9074
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.0916
##
##  McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.5850
##              Specificity : 0.6598
##              Pos Pred Value : 0.9440
##              Neg Pred Value : 0.1396
##              Prevalence : 0.9074
##              Detection Rate : 0.5309
##              Detection Prevalence : 0.5624
##              Balanced Accuracy : 0.6224
##
##              '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')
```



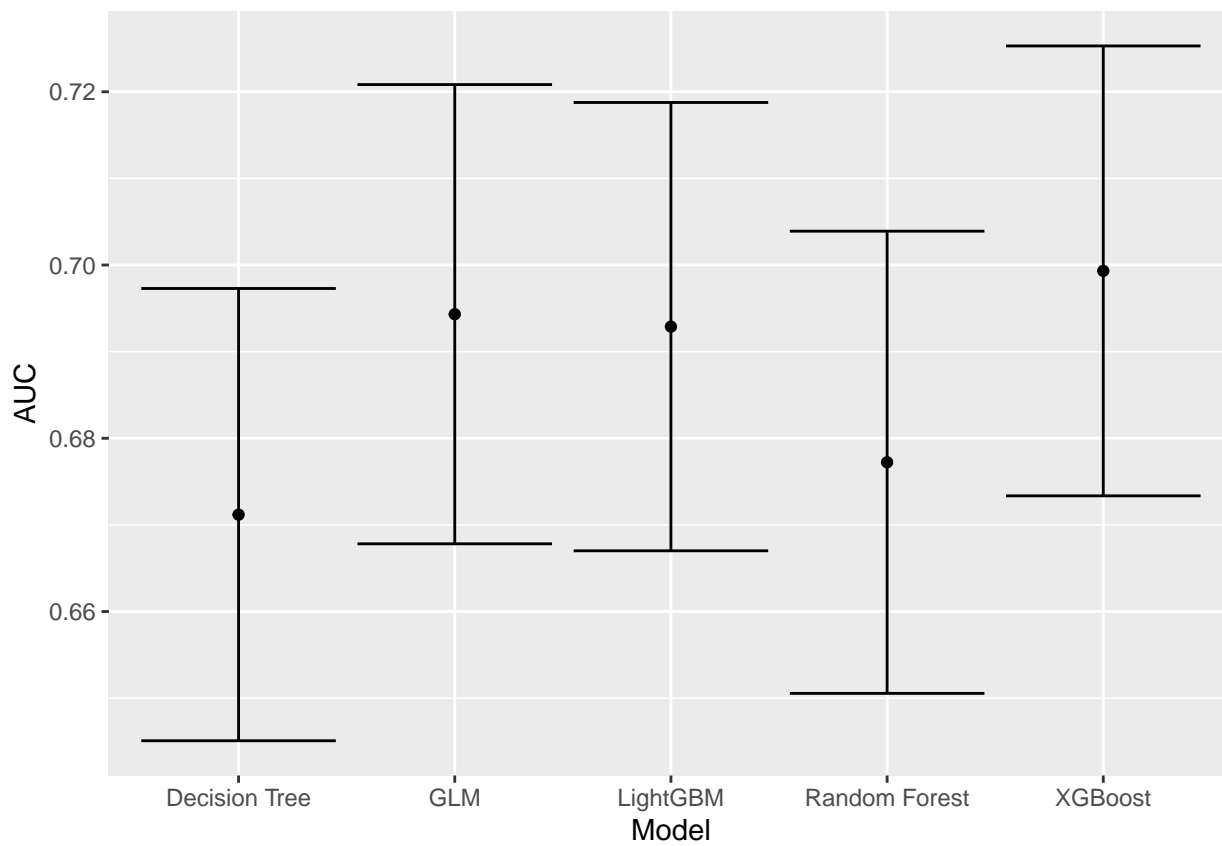
```
# extract_vip(final_rf_fit, pred_wrapper = predict,
#             reference_class = "1", use_matrix = FALSE,
#             method = 'permute')
```

Minutes to run: 69.791

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: 0.003