

Model Selection - readmission_180d

Eduardo Yuki Yada

Global parameters

```
k = 5 # Number of folds for cross validation
grid_size = 15 # Number of parameter combination to tune on each model
```

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")
```

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

```
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))

if (is.null(features_list)) {
  features = eligible_features
} else {
  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. espironolactona
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: 0.002

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(
  mtry = tune(),
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  loss_reduction = tune()
) %>%
  set_engine("xgboost",
             nthread = 8) %>%
  set_mode("classification")

xgboost_grid <- grid_latin_hypercube(
  finalize(mtry(), df_train),
  trees(range = c(100L, 300L)),
  min_n(),
  tree_depth(),
  learn_rate(),
  loss_reduction(),
  size = grid_size
)

xgboost_workflow <-
  workflow() %>%
  add_recipe(xgboost_recipe) %>%
  add_model(xgboost_spec)
```

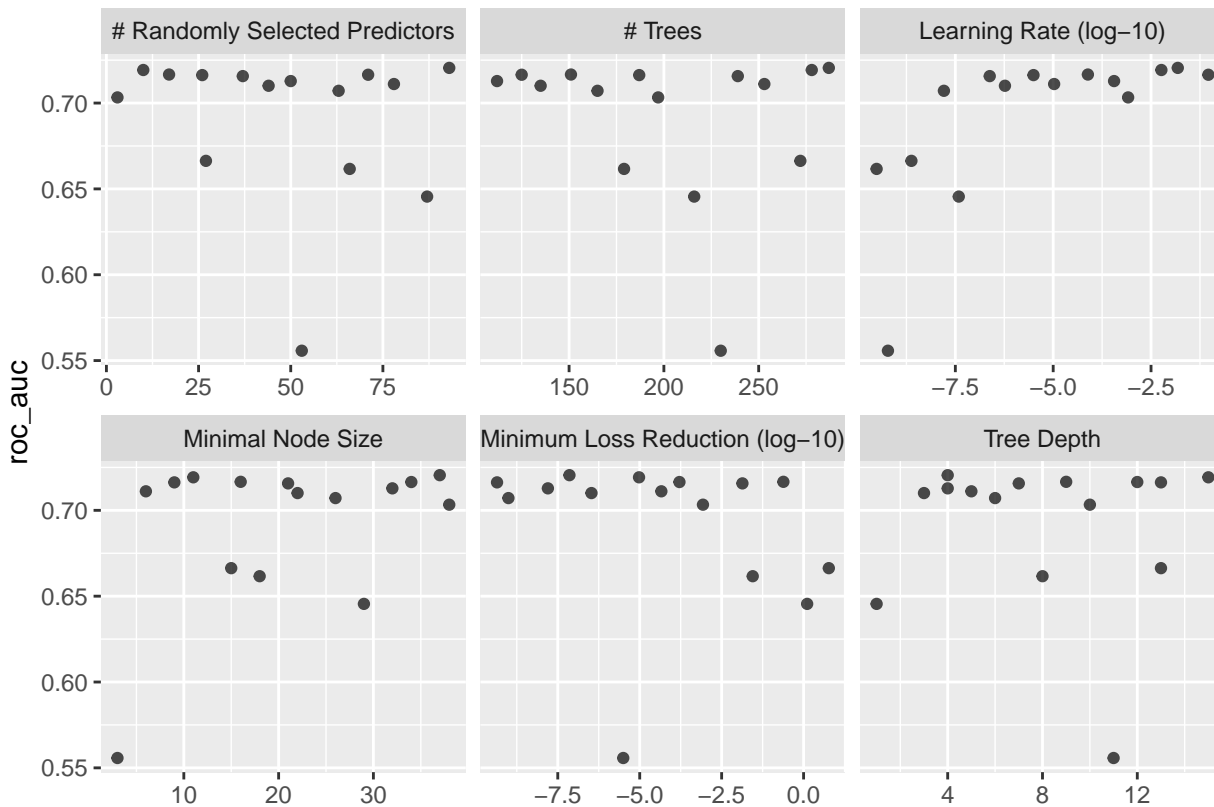
```
xgboost_tune <-
  xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)

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

```
## # A tibble: 5 x 12
##   mtry trees min_n tree_depth learn_rate loss_reduction .metric .estimator mean n std_err .config
##   <int> <int> <int>    <int>    <dbl>    <dbl> <chr>    <chr>    <dbl> <int>  <dbl> <chr>
## 1    93   287    37         4 0.0153      7.18e- 8 roc_auc  binary    0.720     5  0.0132 Preproc~
## 2    10   278    11        15 0.00581     9.55e- 6 roc_auc  binary    0.719     5  0.0114 Preproc~
## 3    17   151    16         9 0.0000770    2.42e- 1 roc_auc  binary    0.717     5  0.0110 Preproc~
## 4    71   125    34        12 0.0923     1.63e- 4 roc_auc  binary    0.716     5  0.0136 Preproc~
## 5    26   187     9        13 0.00000314  4.44e-10 roc_auc  binary    0.716     5  0.0111 Preproc~
```

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

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

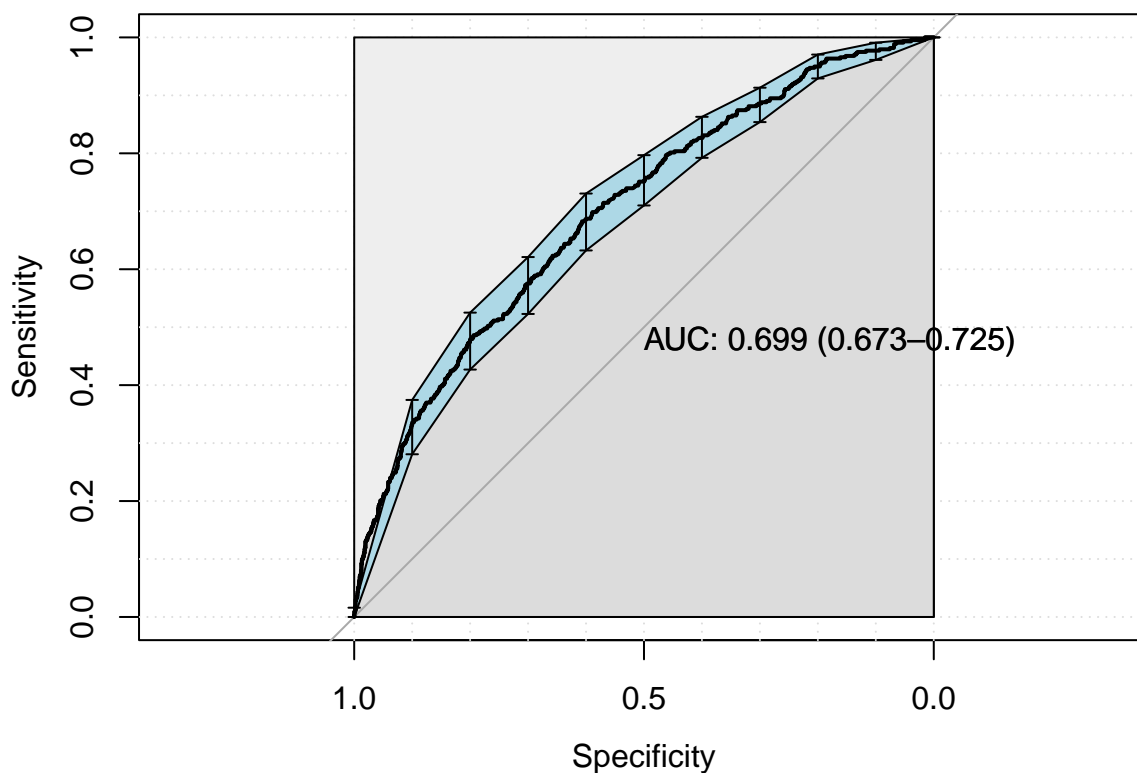


```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

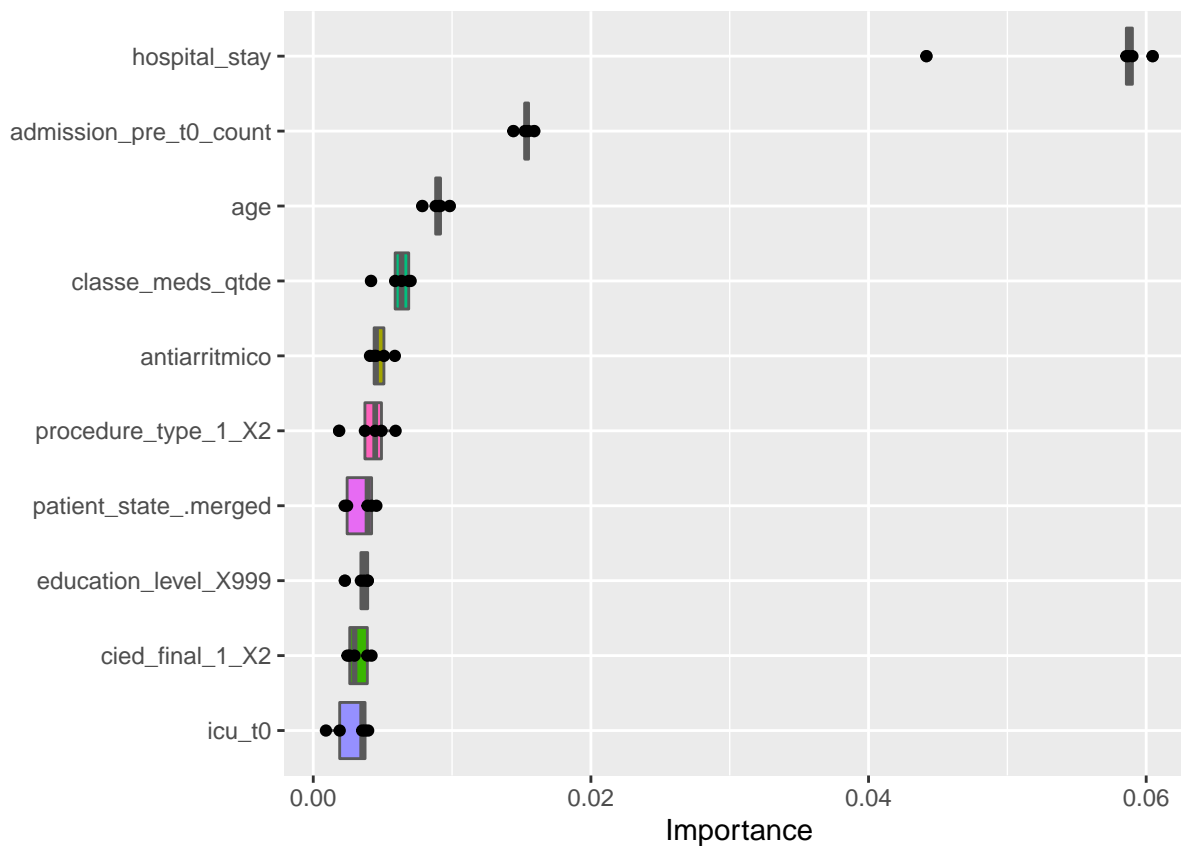
last_xgboost_fit <-
  final_xgboost_workflow %>%
  last_fit(df_split)

final_xgboost_fit <- extract_workflow(last_xgboost_fit)

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



```
## [1] "Optimal Threshold: 0.08"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 2605  139
##    1 1687  299
##
##              Accuracy : 0.614
##              95% CI   : (0.5999, 0.6279)
##    No Information Rate : 0.9074
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.112
##
## Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.6069
##              Specificity : 0.6826
##              Pos Pred Value : 0.9493
##              Neg Pred Value : 0.1506
##              Prevalence : 0.9074
##              Detection Rate : 0.5507
##              Detection Prevalence : 0.5801
##              Balanced Accuracy : 0.6448
##
##              '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(trees, mtry, min_n, tree_depth, learn_rate, loss_reduction) %>%
  as.list

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

Minutes to run: 2.561

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(
  mtry = tune(),
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  loss_reduction = tune(),
  sample_size = 1
) %>%
```

```

set_engine("lightgbm",
           nthread = 8) %>%
set_mode("classification")

lightgbm_grid <- grid_latin_hypercube(
  finalize(mtry(), df_train),
  trees(range = c(100L, 300L)),
  min_n(),
  tree_depth(),
  learn_rate(),
  loss_reduction(),
  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)

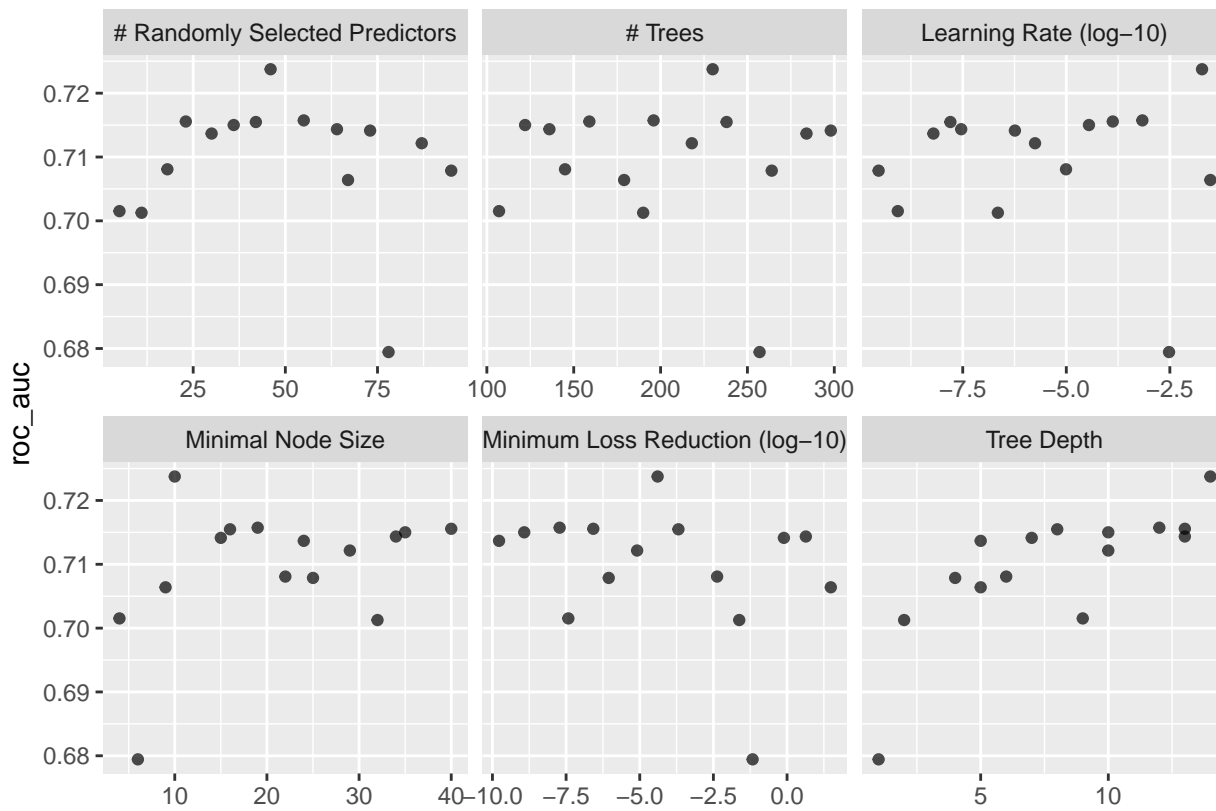
lightgbm_tune %>%
  show_best("roc_auc")

## # A tibble: 5 x 12
##   mtry trees min_n tree_depth   learn_rate loss_reduction .metric .estima~1 mean     n std_err .config
##   <int> <int> <int>     <int>         <dbl>         <dbl> <chr>   <chr>    <dbl> <int>   <dbl> <chr>
## 1    46   230    10         14 0.0189         0.0000405   roc_auc binary    0.724     5 0.0122 Prepro~
## 2    55   196    19         12 0.000686         0.0000000191 roc_auc binary    0.716     5 0.0105 Prepro~
## 3    23   159    40         13 0.000133         0.000000265   roc_auc binary    0.716     5 0.0103 Prepro~
## 4    42   238    16          8 0.0000000159    0.000205     roc_auc binary    0.715     5 0.0105 Prepro~
## 5    36   122    35         10 0.0000353         0.00000000120 roc_auc binary    0.715     5 0.0107 Prepro~
## # ... with abbreviated variable name 1: .estimator

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

autoplot(lightgbm_tune, metric = "roc_auc")

```

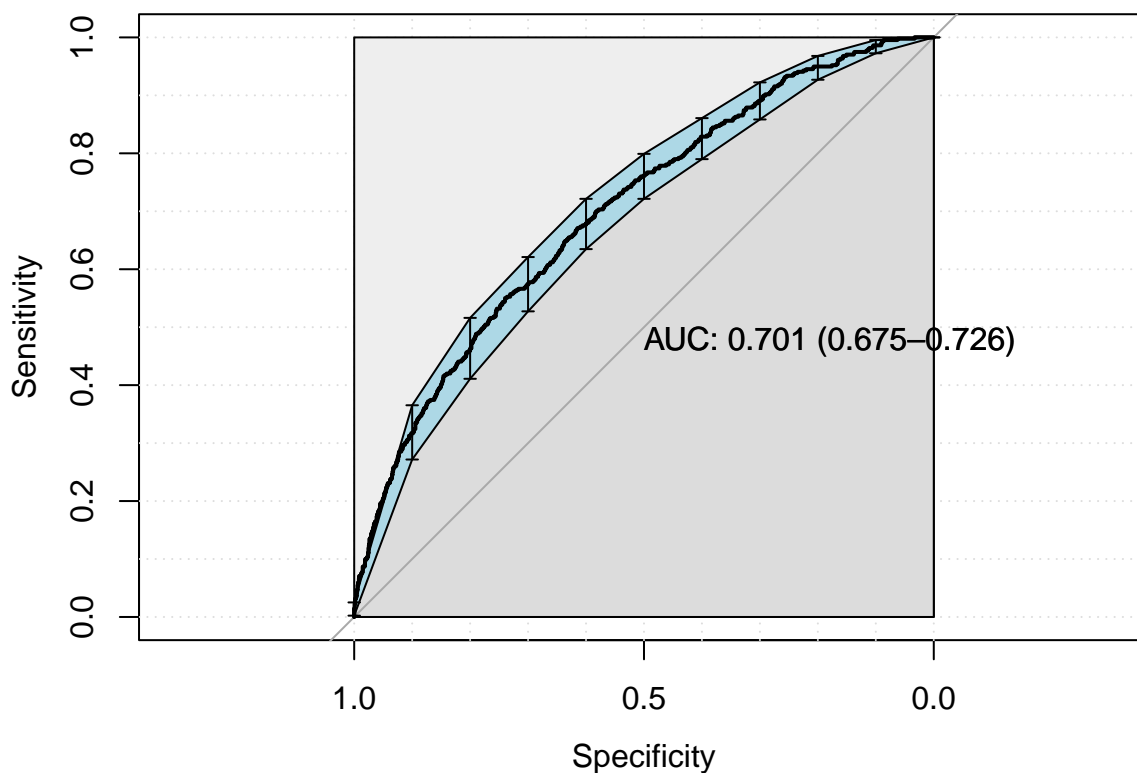



```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

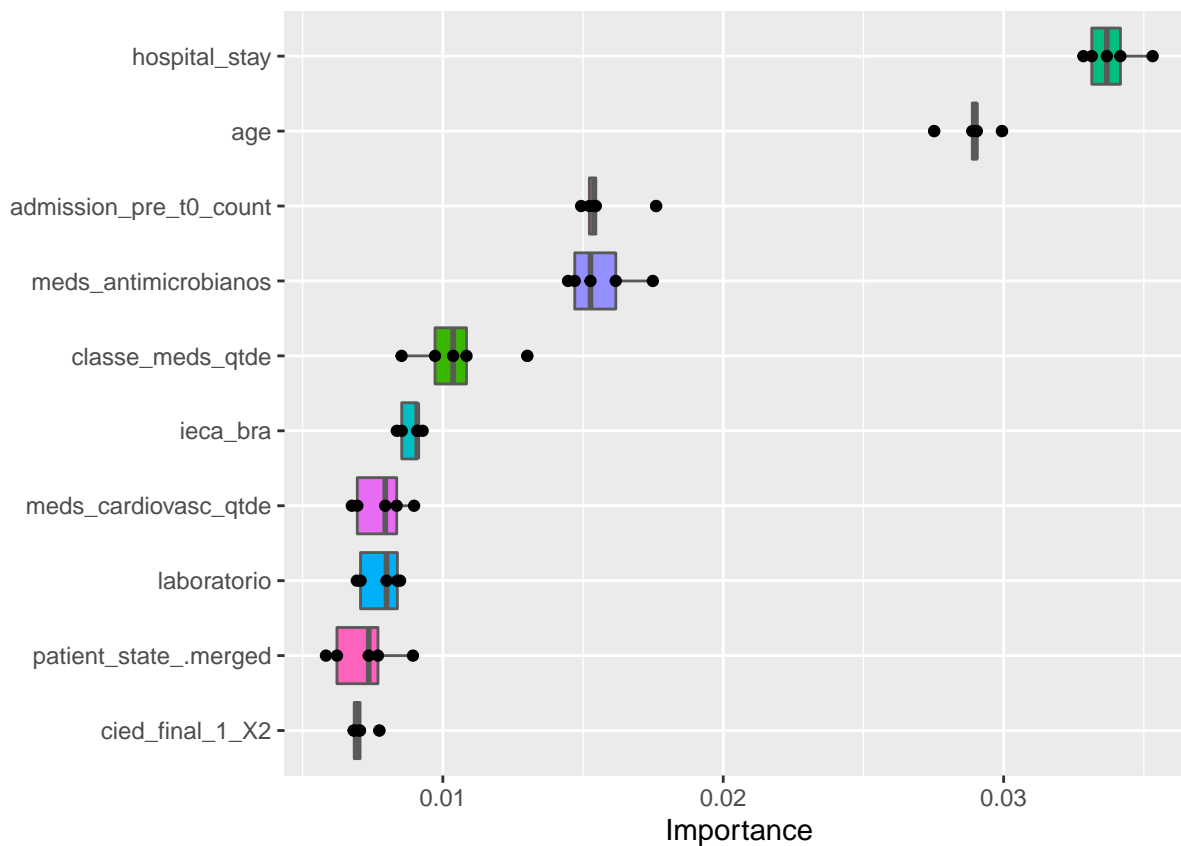
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_auc <- validation(final_lightgbm_fit, df_test)
```



```
## [1] "Optimal Threshold: 0.10"
## Confusion Matrix and Statistics
##
##      reference
## data    0    1
##    0 3175 197
##    1 1117 241
##
##              Accuracy : 0.7222
##              95% CI   : (0.7092, 0.7349)
##    No Information Rate : 0.9074
##    P-Value [Acc > NIR] : 1
##
##              Kappa   : 0.1492
##
## Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.7397
##              Specificity : 0.5502
##              Pos Pred Value : 0.9416
##              Neg Pred Value : 0.1775
##              Prevalence : 0.9074
##              Detection Rate : 0.6712
##              Detection Prevalence : 0.7129
##              Balanced Accuracy : 0.6450
##
##              'Positive' Class : 0
##
pfun_lightgbm <- function(object, newdata) predict(object, data = newdata)
extract_vip(final_lightgbm_fit, pred_wrapper = pfun_lightgbm,
            reference_class = "1")
```



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

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

Minutes to run: 2.892

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 = tune(), mixture = tune()) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

glmnet_grid <- grid_latin_hypercube(penalty(),
  mixture(),
```

```

size = grid_size)

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

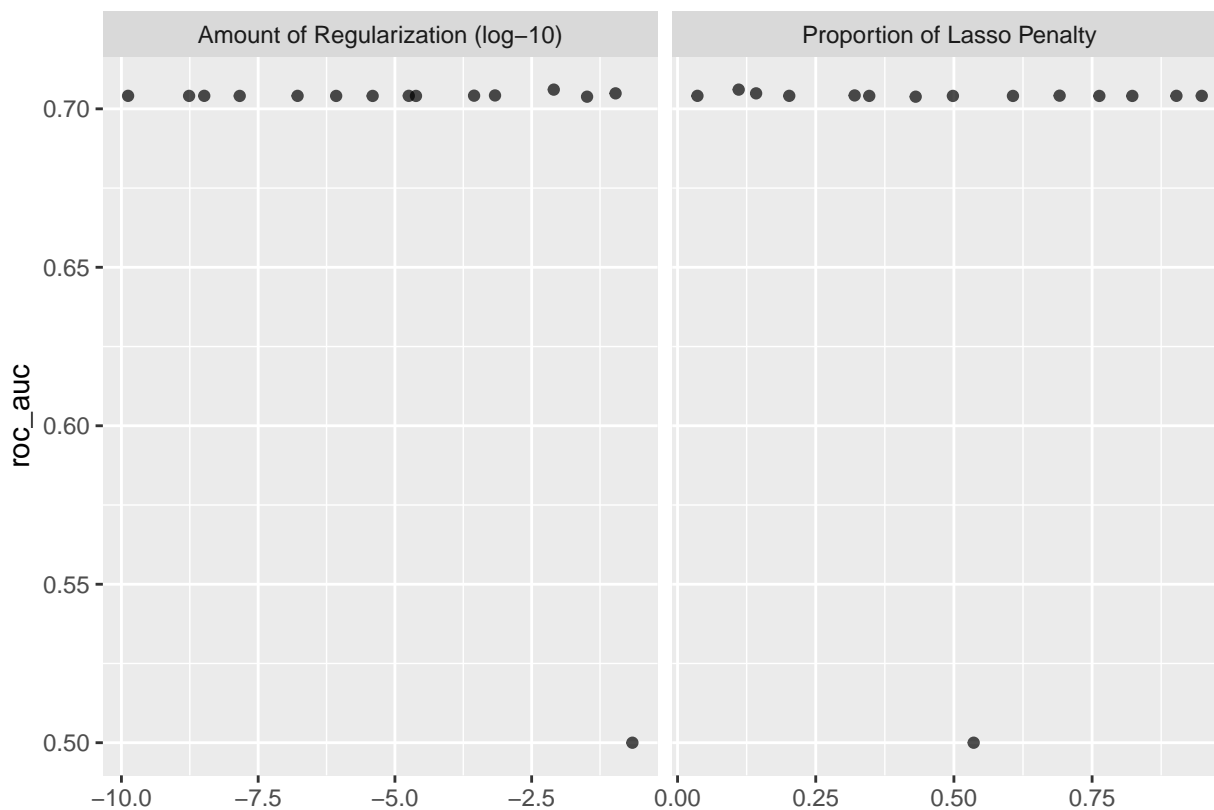
glmnet_tune <-
  glmnet_workflow %>%
  tune_grid(resamples = df_folds,
            grid = glmnet_grid)

glmnet_tune %>%
  collect_metrics()

## # A tibble: 30 x 8
##   penalty mixture .metric .estimator mean      n std_err .config
##   <dbl>   <dbl> <chr>   <chr>   <dbl> <int>   <dbl> <chr>
## 1 1.32e-10 0.0360 accuracy binary    0.906     5 0.00156 Preprocessor1_Model01
## 2 1.32e-10 0.0360 roc_auc   binary    0.704     5 0.0120  Preprocessor1_Model01
## 3 8.03e- 3 0.111 accuracy binary    0.906     5 0.00215 Preprocessor1_Model02
## 4 8.03e- 3 0.111 roc_auc   binary    0.706     5 0.0122  Preprocessor1_Model02
## 5 1.08e- 1 0.142 accuracy binary    0.905     5 0.00154 Preprocessor1_Model03
## 6 1.08e- 1 0.142 roc_auc   binary    0.705     5 0.00971 Preprocessor1_Model03
## 7 1.79e- 5 0.202 accuracy binary    0.906     5 0.00156 Preprocessor1_Model04
## 8 1.79e- 5 0.202 roc_auc   binary    0.704     5 0.0120  Preprocessor1_Model04
## 9 6.77e- 4 0.320 accuracy binary    0.906     5 0.00165 Preprocessor1_Model05
## 10 6.77e- 4 0.320 roc_auc   binary    0.704     5 0.0120  Preprocessor1_Model05
## # ... with 20 more rows
## # i Use 'print(n = ...)' to see more rows

autoplot(glmnet_tune, metric = "roc_auc")

```



```

glmnet_tune %>%
  show_best("roc_auc")

## # A tibble: 5 x 8
##   penalty mixture .metric .estimator mean     n std_err .config
##   <dbl>   <dbl> <chr>   <chr>   <dbl> <int>   <dbl> <chr>
## 1 0.00803   0.111 roc_auc binary    0.706     5 0.0122 Preprocessor1_Model02
## 2 0.108     0.142 roc_auc binary    0.705     5 0.00971 Preprocessor1_Model03
## 3 0.000677 0.320 roc_auc binary    0.704     5 0.0120 Preprocessor1_Model05
## 4 0.000280 0.691 roc_auc binary    0.704     5 0.0120 Preprocessor1_Model11
## 5 0.0000179 0.202 roc_auc binary    0.704     5 0.0120 Preprocessor1_Model04

best_glmnet <- glmnet_tune %>%
  select_best("roc_auc")

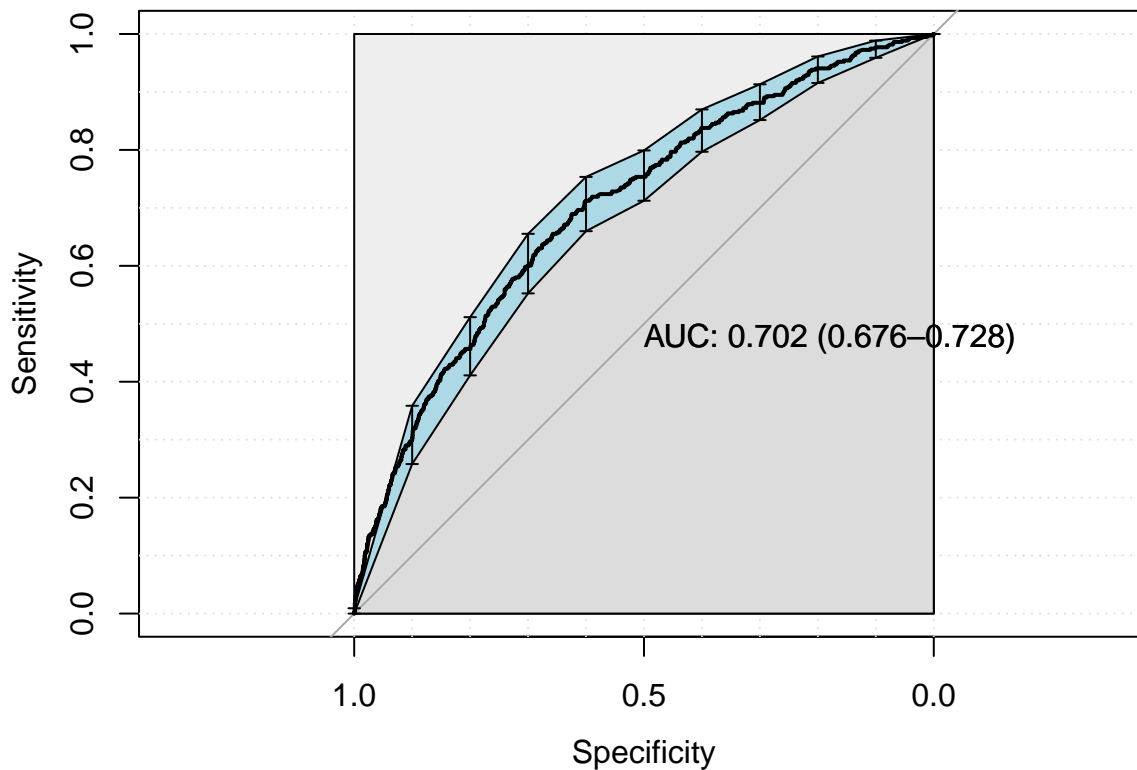
final_glmnet_workflow <-
  glmnet_workflow %>%
  finalize_workflow(best_glmnet)

last_glmnet_fit <-
  final_glmnet_workflow %>%
  last_fit(df_split)

final_glmnet_fit <- extract_workflow(last_glmnet_fit)

glmnet_auc <- validation(final_glmnet_fit, df_test)

```



```

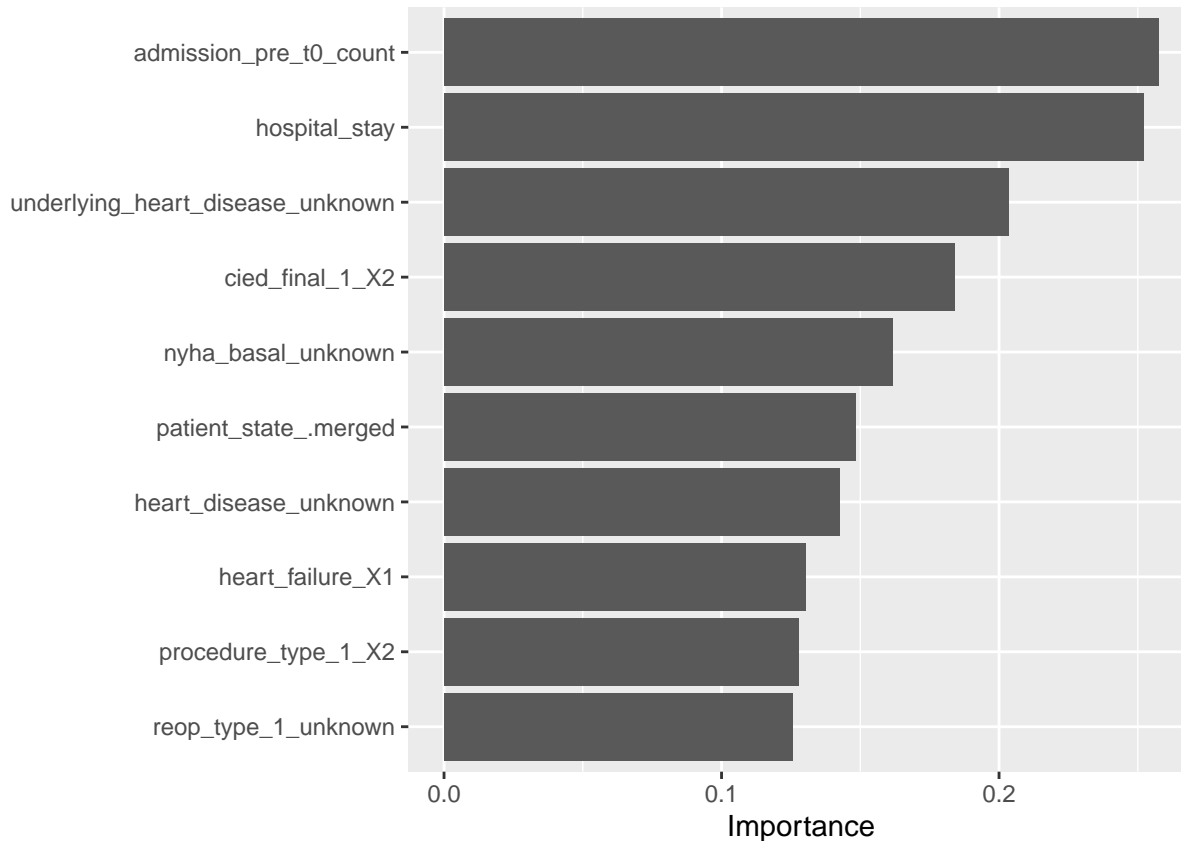
## [1] "Optimal Threshold: 0.08"
## Confusion Matrix and Statistics
##
##   reference
## data      0      1
## 0 2681 136
## 1 1611 302

```

```
##
##           Accuracy : 0.6307
##           95% CI : (0.6167, 0.6444)
##           No Information Rate : 0.9074
##           P-Value [Acc > NIR] : 1
##
##           Kappa : 0.1251
##
##  McNemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.6247
##           Specificity : 0.6895
##           Pos Pred Value : 0.9517
##           Neg Pred Value : 0.1579
##           Prevalence : 0.9074
##           Detection Rate : 0.5668
##           Detection Prevalence : 0.5956
##           Balanced Accuracy : 0.6571
##
##           'Positive' Class : 0
##
```

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

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



Minutes to run:

1.184

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)

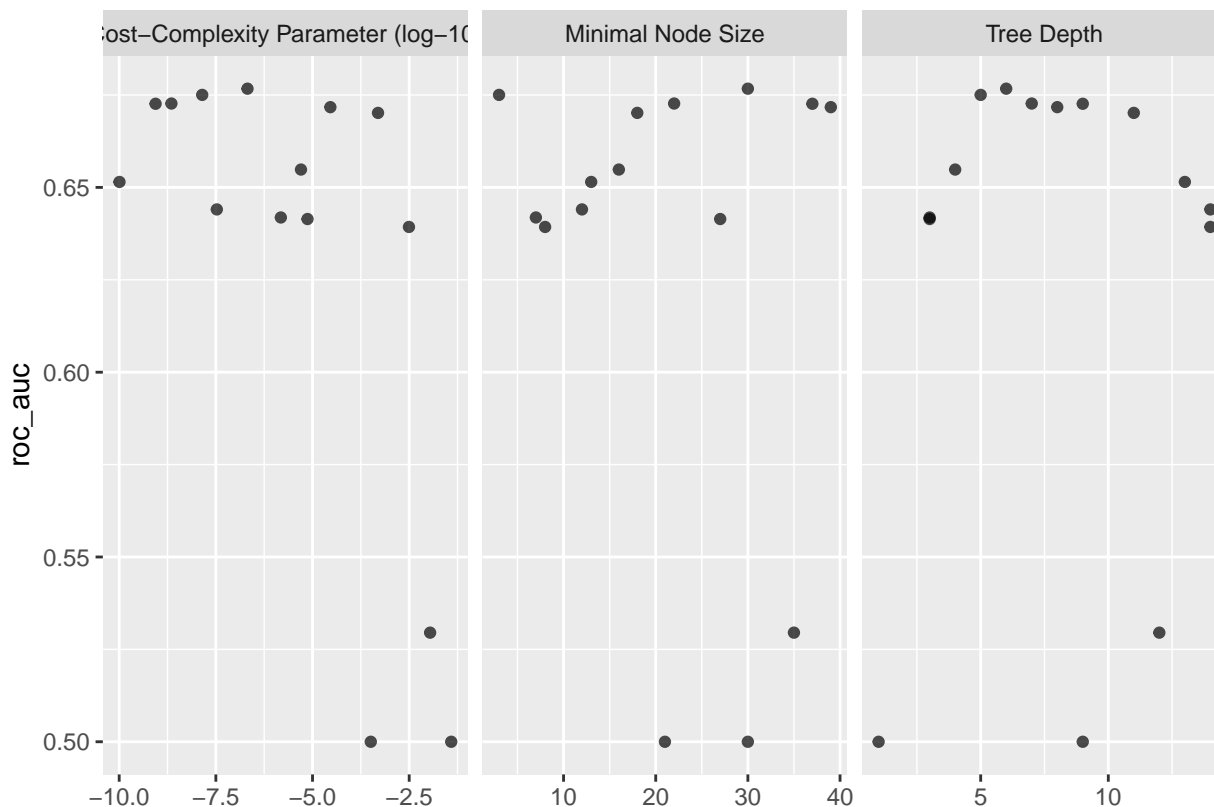
tree_workflow <-
  workflow() %>%
  add_recipe(tree_recipe) %>%
  add_model(tree_spec)

tree_tune <-
  tree_workflow %>%
  tune_grid(resamples = df_folds,
            grid = tree_grid)

tree_tune %>%
  collect_metrics()

autoplot(tree_tune, metric = "roc_auc")

```



```

tree_tune %>%
  show_best("roc_auc")

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

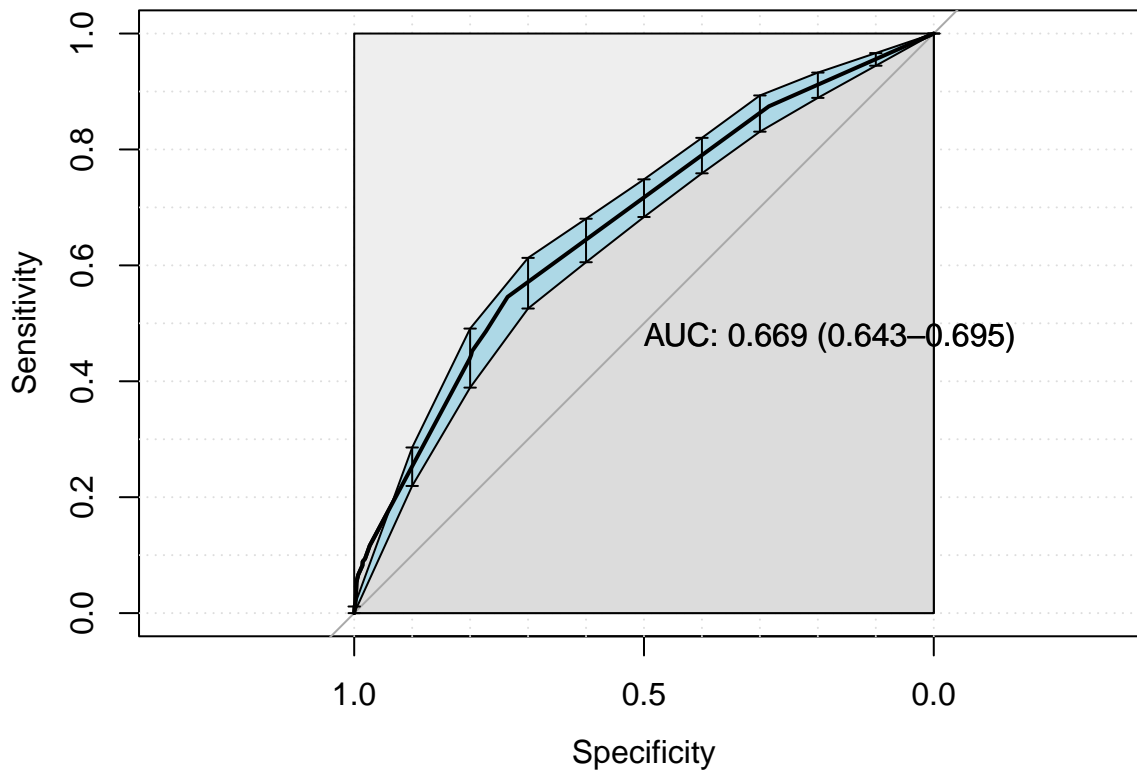
final_tree_workflow <-
  tree_workflow %>%
  finalize_workflow(best_tree)

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)

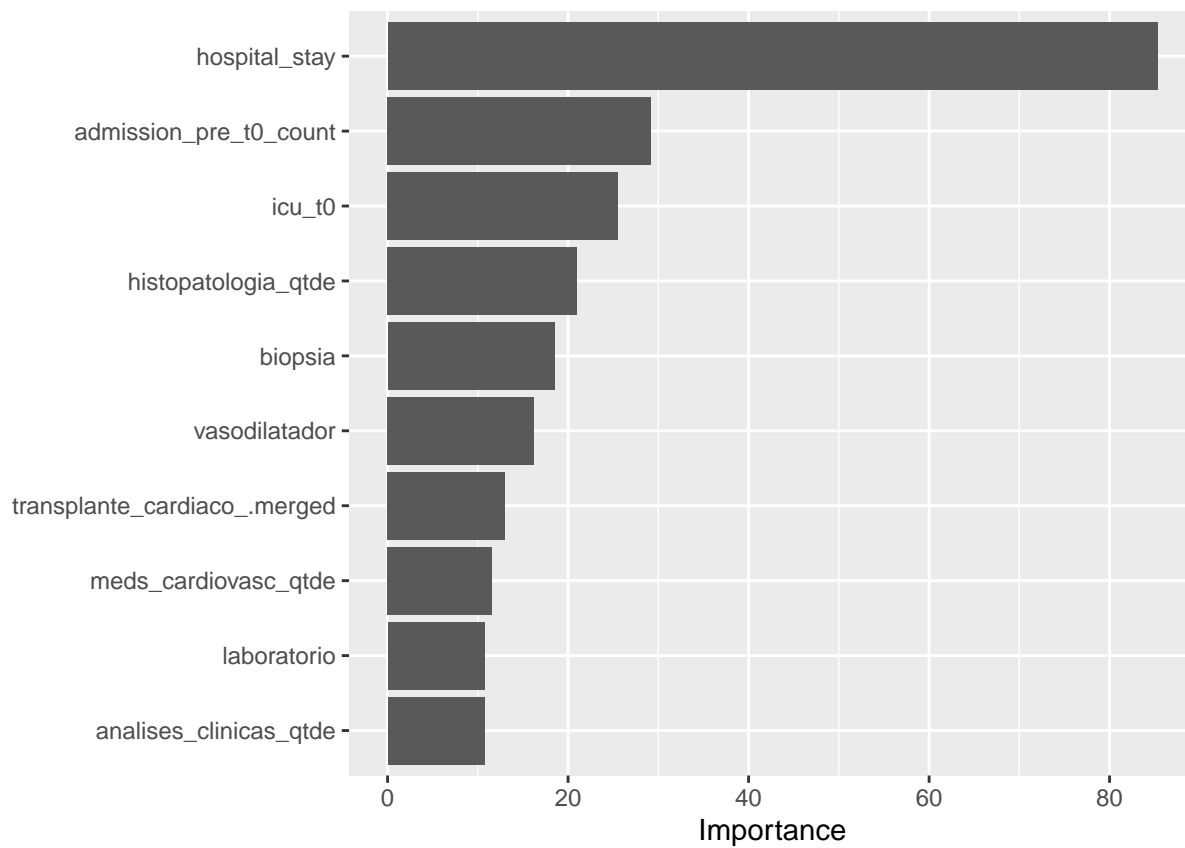
```



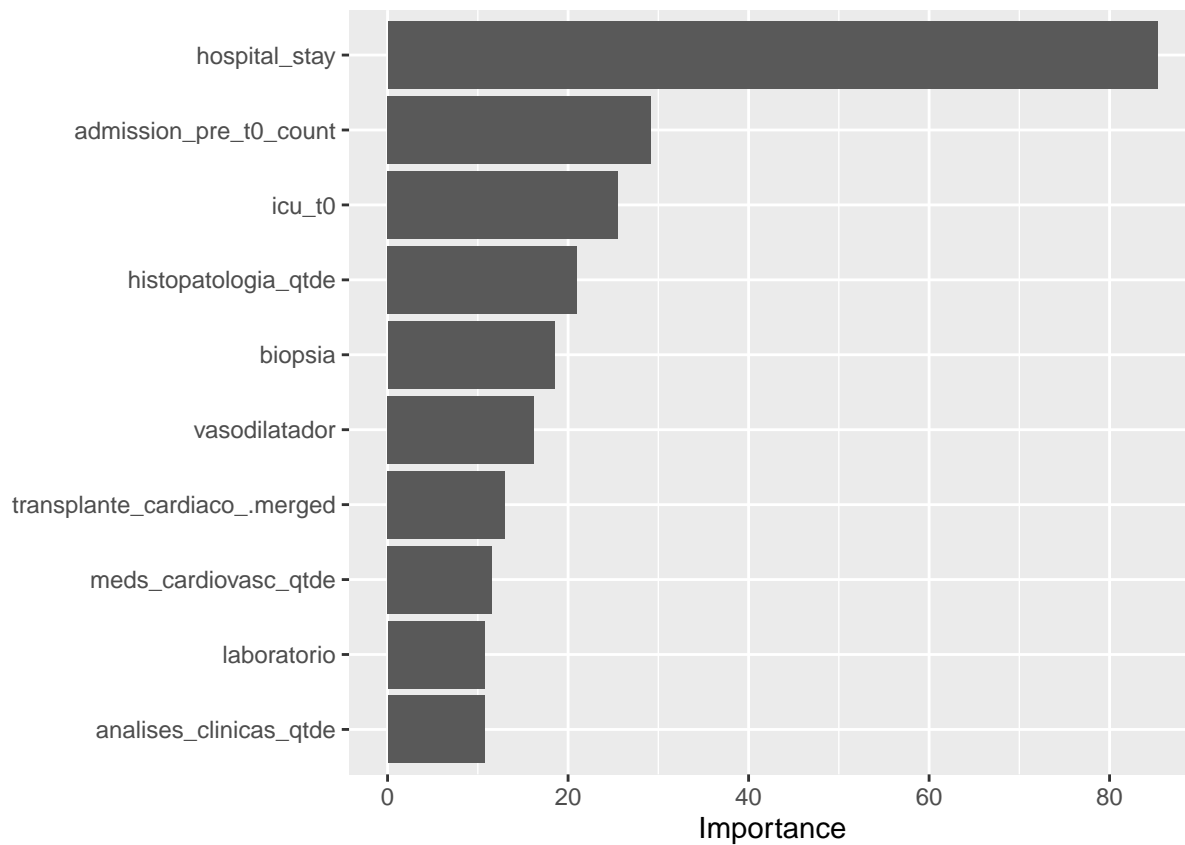
```

if (tree_auc$auc > 0.55) {
  final_tree_fit %>%
    extract_fit_parsnip() %>%
    vip()
}

```

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

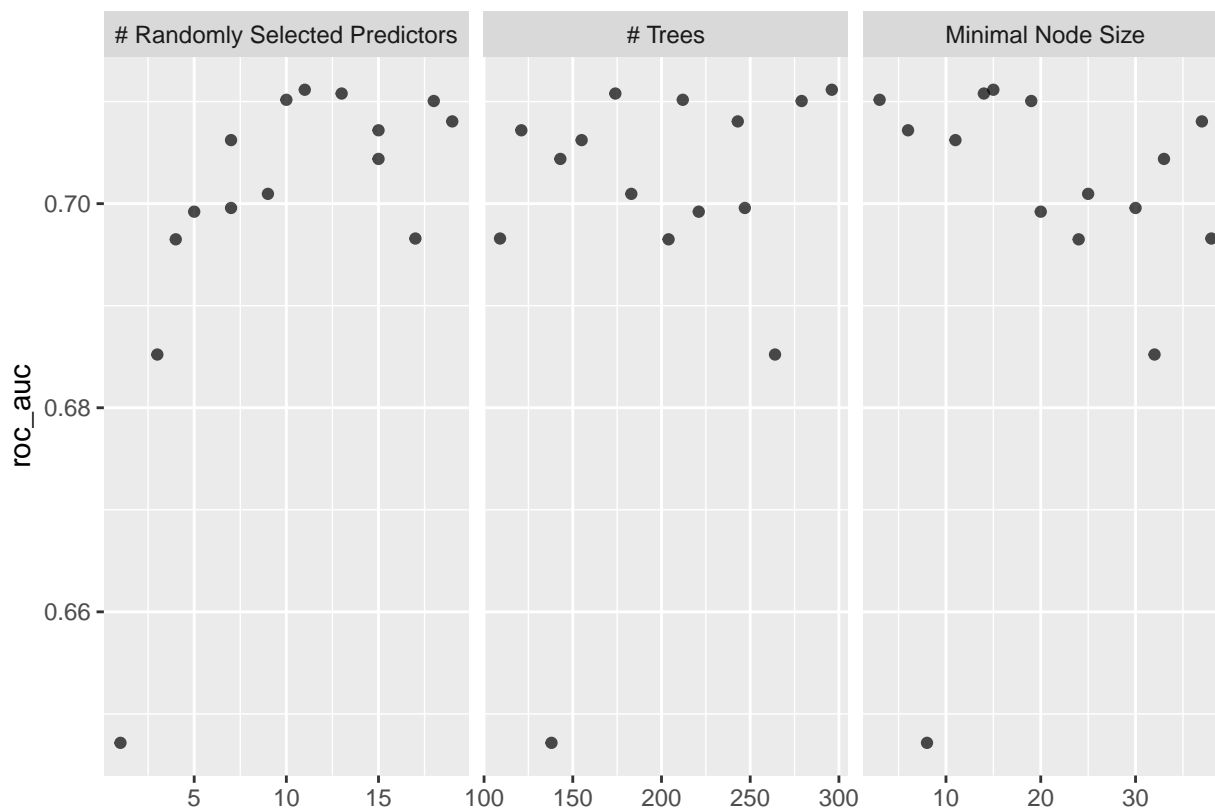


1.889

Minutes to run:

Random Forest

```
rf_recipe <-  
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,  
    data = df_train) %>%  
  step_nominal(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, 20L)),  
  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()  
  
autoplot(rf_tune, metric = "roc_auc")
```



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

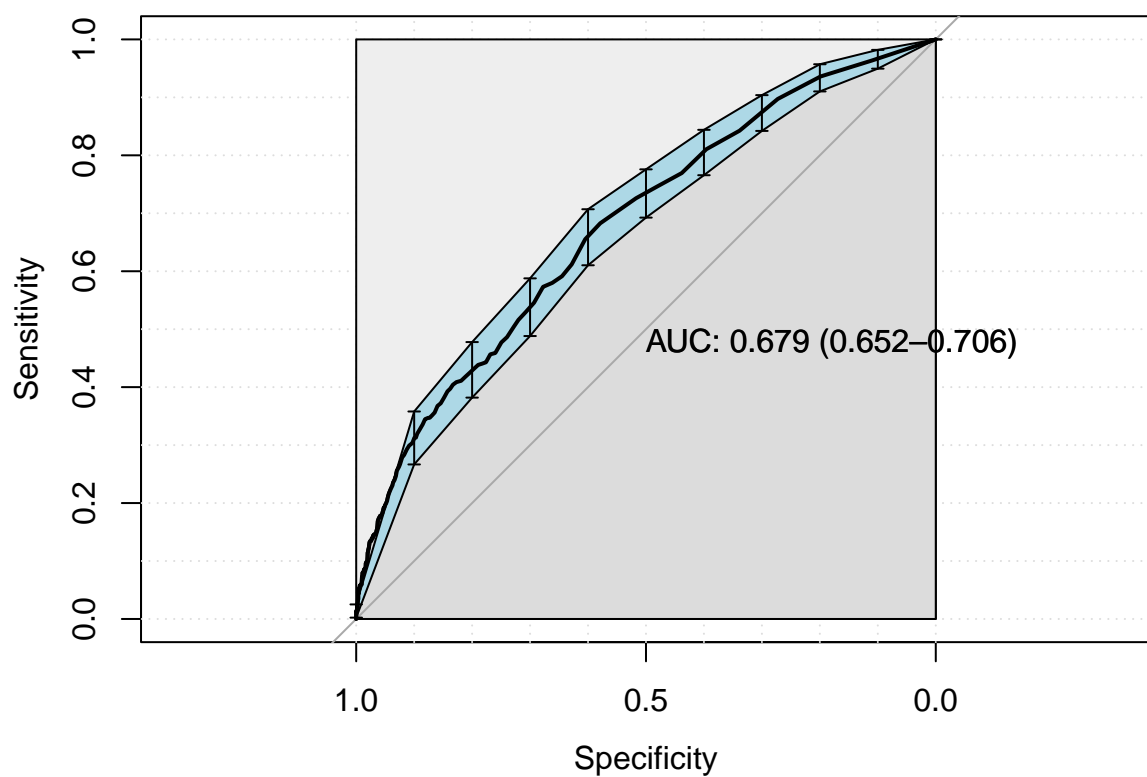
best_rf <- rf_tune %>%
  select_best("roc_auc")

final_rf_workflow <-
  rf_workflow %>%
  finalize_workflow(best_rf)

last_rf_fit <-
  final_rf_workflow %>%
  last_fit(df_split)

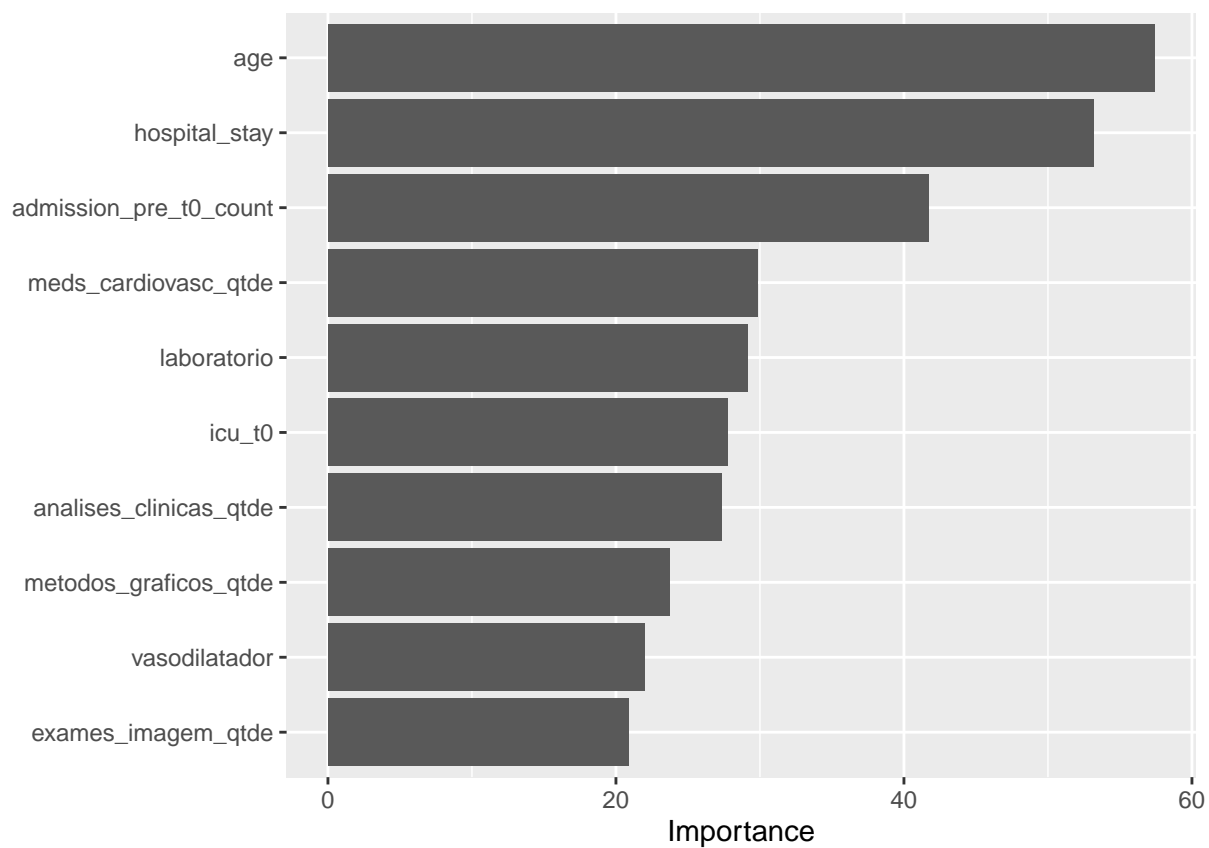
final_rf_fit <- extract_workflow(last_rf_fit)

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')
```



KNN

```
# knn_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())  
#  
# knn_spec <-  
#   nearest_neighbor(neighbors = tune(),  
#                     weight_func = tune(),  
#                     dist_power = tune()) %>%  
#   set_mode("classification") %>%  
#   set_engine("kknn")  
#  
# knn_grid <- grid_latin_hypercube(neighbors(),  
#                                   weight_func(),  
#                                   dist_power(),  
#                                   size = grid_size)  
#  
# knn_workflow <-  
#   workflow() %>%  
#   add_recipe(knn_recipe) %>%  
#   add_model(knn_spec)  
#  
# knn_tune <-  
#   knn_workflow %>%  
#   tune_grid(resamples = df_folds,  
#             grid = knn_grid)  
#  
# knn_tune %>%  
#   collect_metrics()  
#  
# autoplot(knn_tune, metric = "roc_auc")  
#  
# knn_tune %>%  
#   show_best("roc_auc")  
#  
# best_knn <- knn_tune %>%  
#   select_best("roc_auc")  
#  
# final_knn_workflow <-  
#   knn_workflow %>%  
#   finalize_workflow(best_knn)  
#  
# last_knn_fit <-  
#   final_knn_workflow %>%  
#   last_fit(df_split)  
#  
# final_knn_fit <- extract_workflow(last_knn_fit)  
#  
# knn_auc = validation(final_knn_fit, df_test)
```

Minutes to run: 0

SVM

```
# sum_recipe <-
#   recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
#   step_nominal(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())
#
# sum_spec <-
#   sum_rbf(cost = tune(), rbf_sigma = tune()) %>%
#   set_mode("classification") %>%
#   set_engine("kernlab")
#
# sum_grid <- grid_latin_hypercube(cost(),
#                                   rbf_sigma(),
#                                   size = grid_size)
#
# sum_workflow <-
#   workflow() %>%
#   add_recipe(sum_recipe) %>%
#   add_model(sum_spec)
#
# sum_tune <-
#   sum_workflow %>%
#   tune_grid(resamples = df_folds,
#             grid = grid_size)
#
# sum_tune %>%
#   collect_metrics()
#
# autoplot(sum_tune, metric = "roc_auc")
#
# sum_tune %>%
#   show_best("roc_auc")
#
# best_sum <- sum_tune %>%
#   select_best("roc_auc")
#
# final_sum_workflow <-
#   sum_workflow %>%
#   finalize_workflow(best_sum)
#
# last_sum_fit <-
#   final_sum_workflow %>%
#   last_fit(df_split)
#
# final_sum_fit <- extract_workflow(last_sum_fit)
#
# sum_auc = validation(final_sum_fit, df_test)
```

Minutes to run: 0

Models Comparison

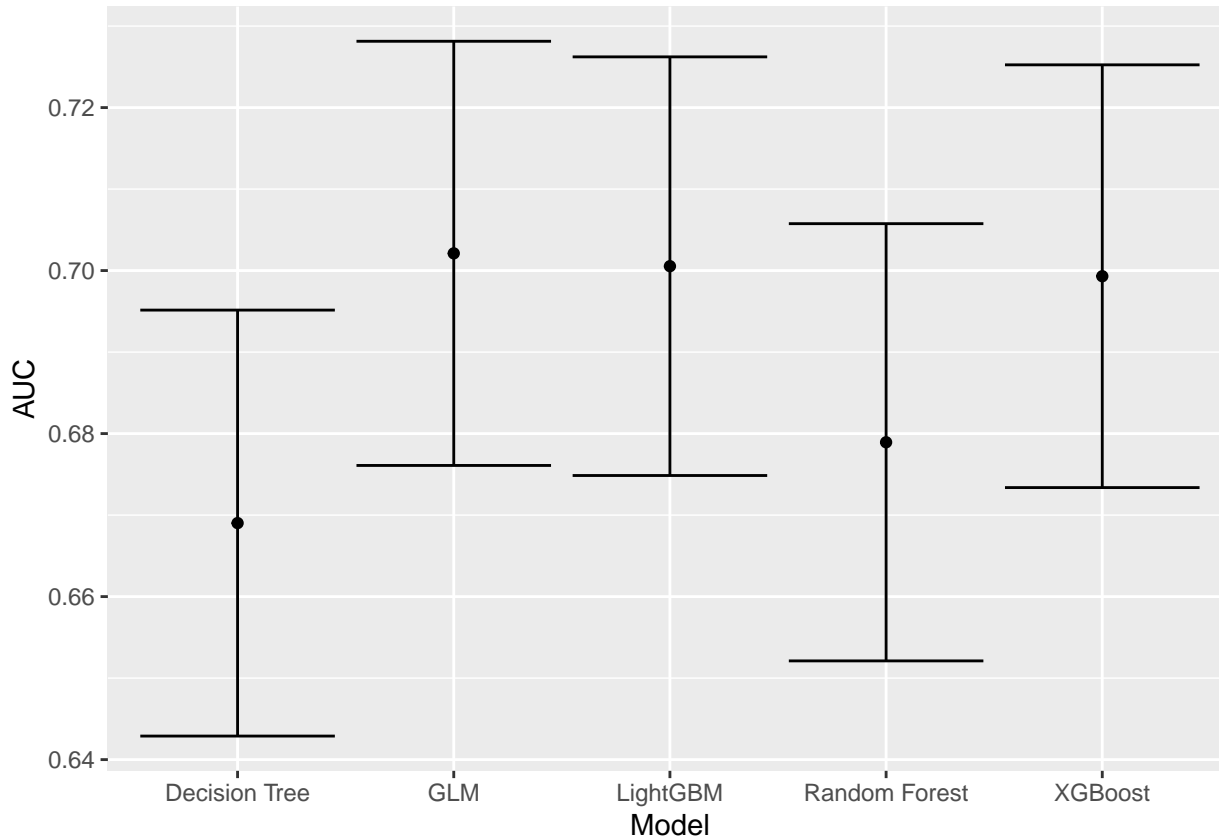
```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'XGBoost', as.numeric(xgboost_auc$auc), xgboost_auc$ci[1], xgboost_auc$ci[3],
```

```

'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)

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