Model Selection - readmission_30d

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: 0.013

```
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. education_level
## 02. underlying_heart_disease
## 03. heart_disease
## 04. nyha_basal
## 05. prior_mi
## 06. heart_failure
## 07. transplant
## 08. endocardites
## 09. hemodialysis
## 10. comorbidities_count
## 11. procedure_type_1
## 12. reop_type_1
## 13. procedure_type_new
## 14. cied_final_1
## 15. cied_final_group_1
## 16. admission_pre_t0_count
## 17. admission_pre_t0_180d
## 18. icu_t0
## 19. dialysis_t0
## 20. admission_t0_emergency
## 21. aco
## 22. antiarritmico
## 23. betabloqueador
## 24. ieca_bra
## 25. dva
## 26. digoxina
## 27. estatina
## 28. diuretico
## 29. vasodilatador
## 30. insuf_cardiaca
## 31. espironolactona
```

```
## 32. bloq_calcio
## 33. antiplaquetario_ev
## 34. insulina
## 35. anticonvulsivante
## 36. psicofarmacos
## 37. antifungico
## 38. antiviral
## 39. classe_meds_qtde
## 40. meds_cardiovasc_qtde
## 41. meds_antimicrobianos
## 42. ventilacao_mecanica
## 43. cec
## 44. transplante_cardiaco
## 45. outros_proced_cirurgicos
## 46. icp
## 47. intervencao_cv
## 48. cateterismo
## 49. eletrofisiologia
## 50. cateter_venoso_central
## 51. proced_invasivos_qtde
## 52. cve_desf
## 53. transfusao
## 54. equipe_multiprof
## 55. holter
## 56. metodos_graficos_qtde
## 57. laboratorio
## 58. cultura
## 59. analises_clinicas_qtde
## 60. citologia
## 61. biopsia
## 62. histopatologia_qtde
## 63. angio_rm
## 64. angio_tc
## 65. cintilografia
## 66. ecocardiograma
## 67. endoscopia
## 68. flebografia
## 69. pet_ct
## 70. ultrassom
## 71. tomografia
## 72. ressonancia
## 73. exames_imagem_qtde
## 74. bic
## 75. mpp
## 76. 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: 0.004

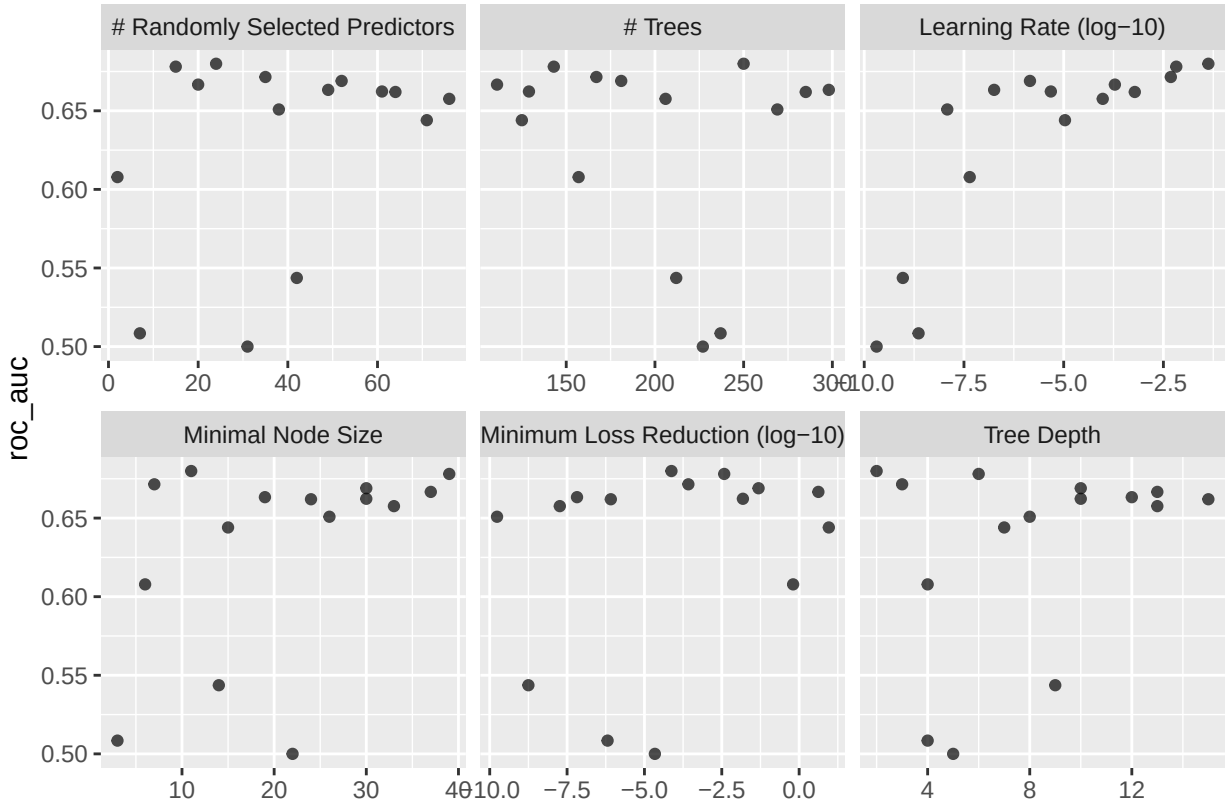
Boosted Tree (XGBoost)

```
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")
```

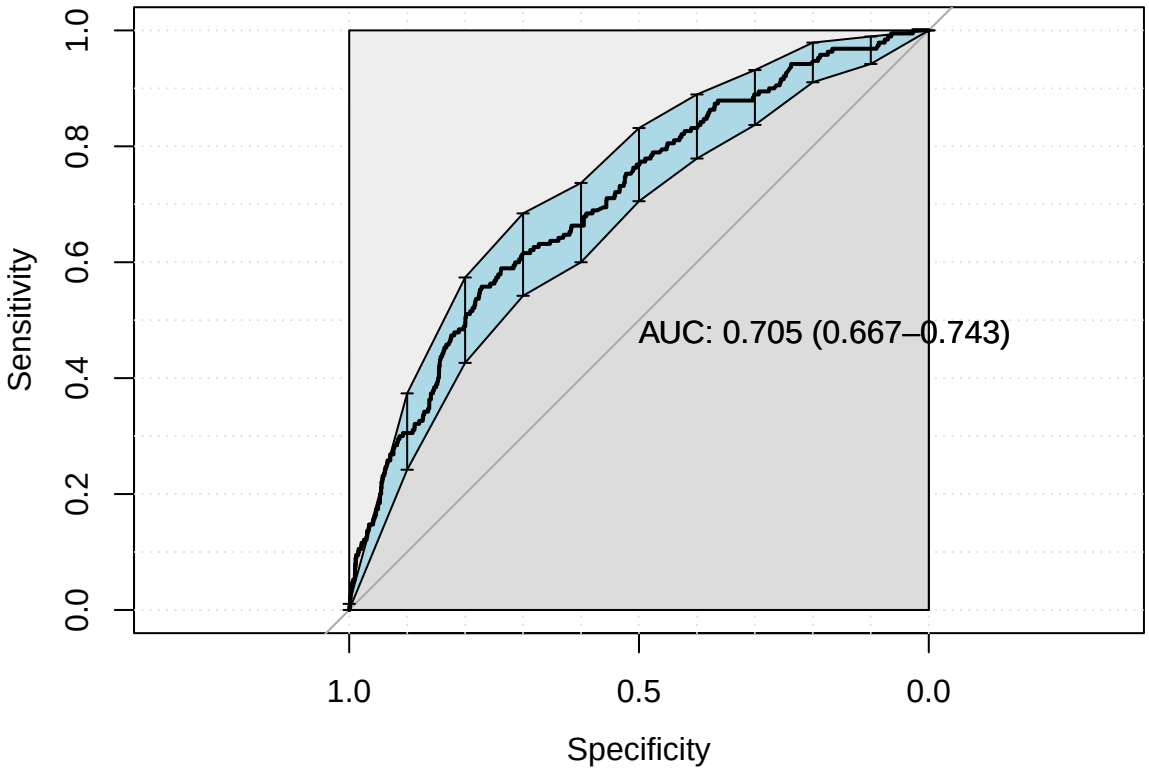
```
## # A tibble: 5 x 12
##    mtry trees min_n tree_depth learn_rate loss_reduction .metric .estimator  mean     n std_err .config
##   <int> <int> <int>      <int>      <dbl>          <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    24   250    11          2 0.0424          0.0000746 roc_auc binary     0.680     5 0.0104  Preproc~
## 2    15   143    39          6 0.00665         0.00379   roc_auc binary     0.678     5 0.00831 Preproc~
## 3    35   167     7          3 0.00485         0.000263  roc_auc binary     0.672     5 0.00616 Preproc~
## 4    52   181    30         10 0.00000143      0.0488    roc_auc binary     0.669     5 0.00542 Preproc~
## 5    20   111    37         13 0.000192        4.19      roc_auc binary     0.667     5 0.00673 Preproc~
```

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

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

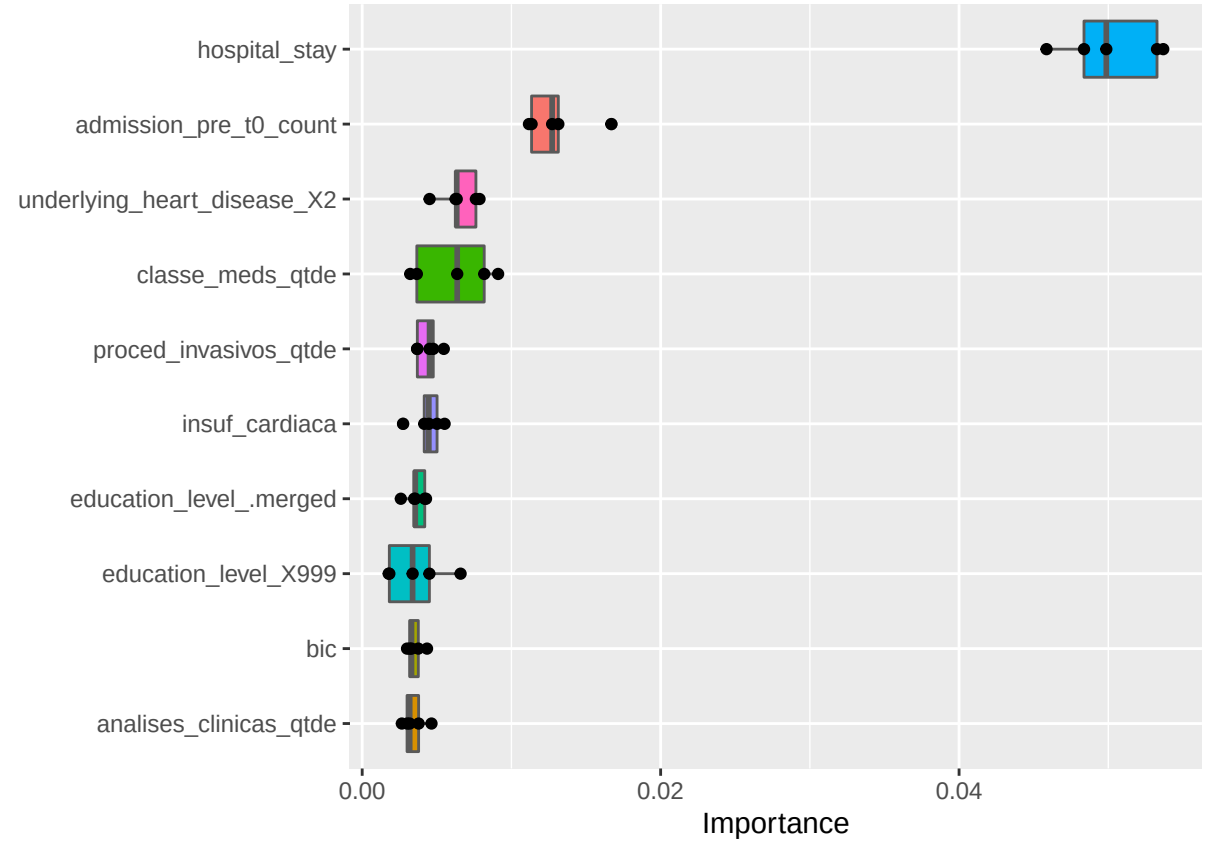



```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

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: 0.05"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3503   84
##    1 1037  106
##
##                Accuracy : 0.763
##                  95% CI : (0.7506, 0.7751)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0968
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77159
##             Specificity : 0.55789
##          Pos Pred Value : 0.97658
##          Neg Pred Value : 0.09274
##              Prevalence : 0.95983
##          Detection Rate : 0.74059
##    Detection Prevalence : 0.75835
##       Balanced Accuracy : 0.66474
##
##        '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: 1.761

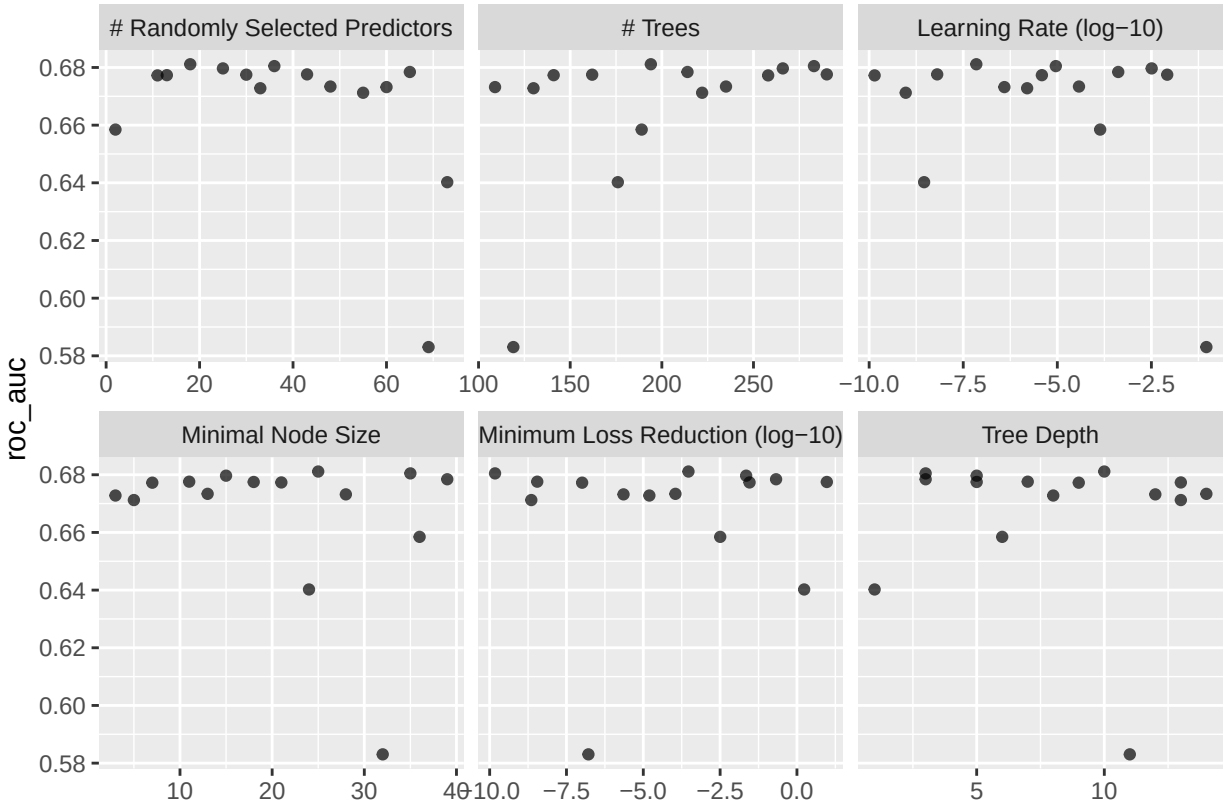
Boosted Tree (LightGBM)

```
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
) %>%
```

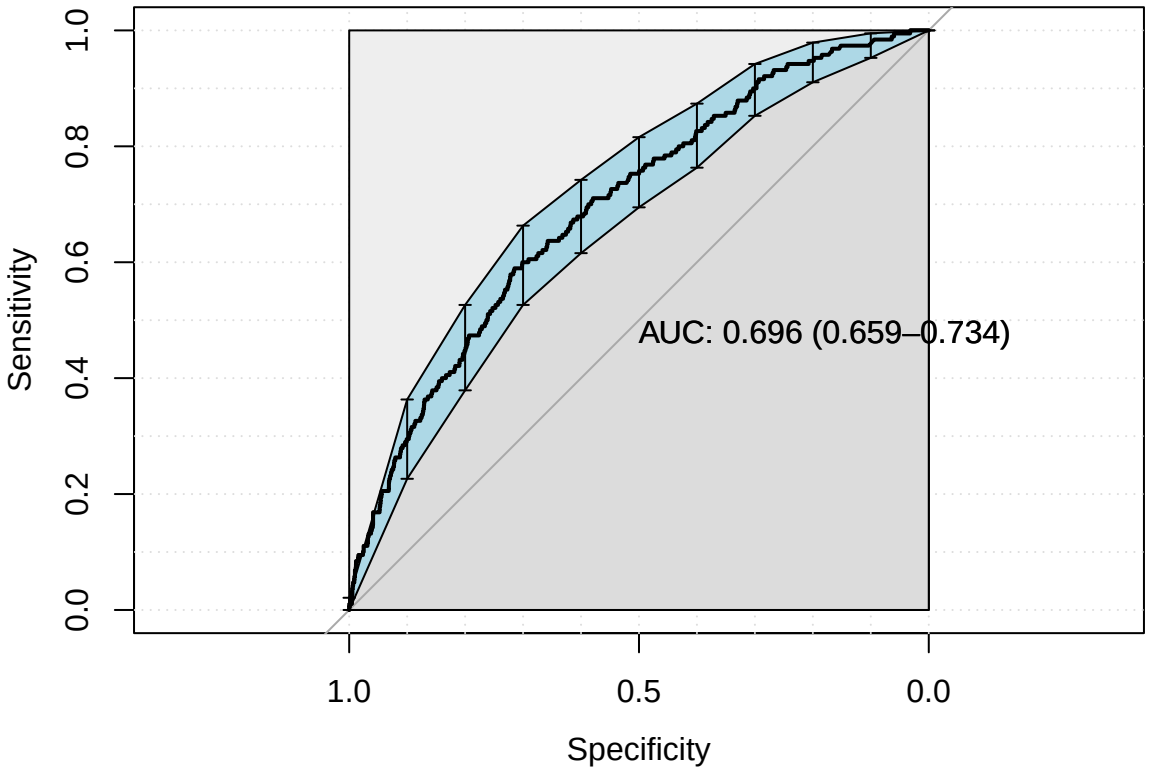
```
  set_engine("lightgbm",
             nthread = 8) %>%
  set_mode("classification")

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 .estim~1  mean     n std_err .config
##   <int> <int> <int>      <int>         <dbl>          <dbl> <chr>   <chr>    <dbl> <int>   <dbl> <chr>
## 1    18   194    25         10 0.0000000704        2.95e- 4 roc_auc binary   0.681     5 0.00924 Prepro~
## 2    36   283    35          3 0.00000913          1.50e-10 roc_auc binary   0.680     5 0.0103  Prepro~
## 3    25   266    15          5 0.00322             2.23e- 2 roc_auc binary   0.680     5 0.00705 Prepro~
## 4    65   214    39          3 0.000416            2.09e- 1 roc_auc binary   0.678     5 0.0101  Prepro~
## 5    43   290    11          7 0.00000000642       3.54e- 9 roc_auc binary   0.678     5 0.00732 Prepro~
## # ... with abbreviated variable name 1: .estimator
```

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

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

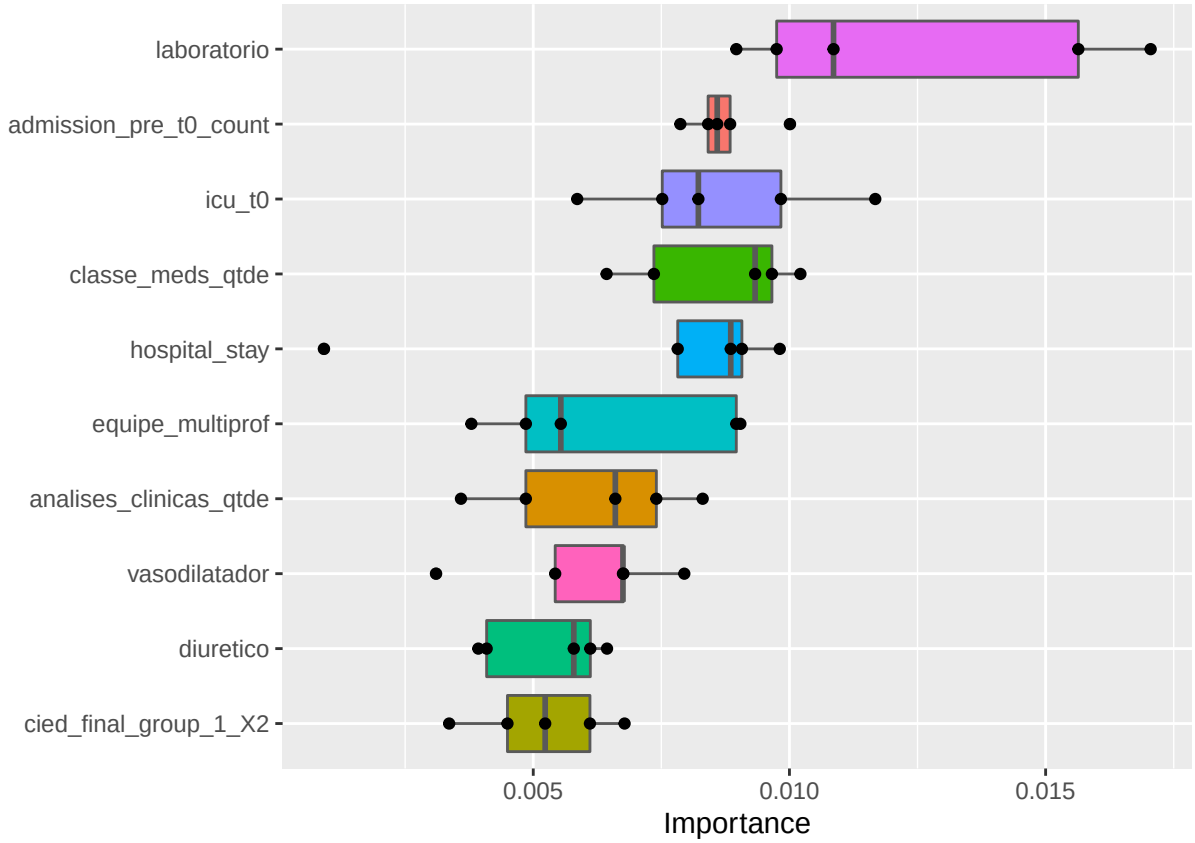



```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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: 0.04"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3246   78
##    1 1294  112
##
##                Accuracy : 0.7099
##                  95% CI : (0.6968, 0.7228)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0749
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71498
##             Specificity : 0.58947
##          Pos Pred Value : 0.97653
##          Neg Pred Value : 0.07966
##              Prevalence : 0.95983
##          Detection Rate : 0.68626
##    Detection Prevalence : 0.70275
##       Balanced Accuracy : 0.65223
##
##        '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: 2.614

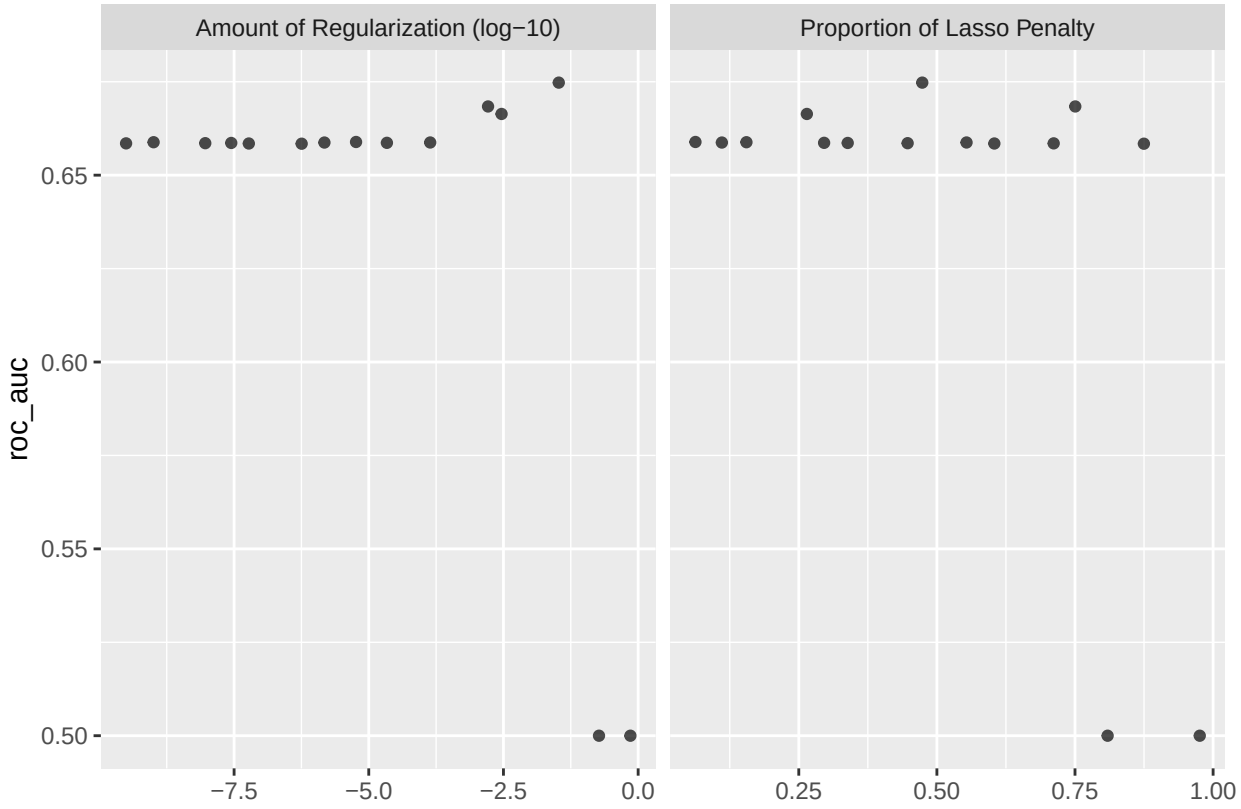
GLM

```
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 0.00000581     0.0629 accuracy binary     0.962     5 0.00231 Preprocessor1_Model01
##  2 0.00000581     0.0629 roc_auc  binary     0.659     5 0.0103  Preprocessor1_Model01
##  3 0.00000150     0.111  accuracy binary     0.962     5 0.00231 Preprocessor1_Model02
##  4 0.00000150     0.111  roc_auc  binary     0.659     5 0.0103  Preprocessor1_Model02
##  5 0.00000000101  0.155  accuracy binary     0.962     5 0.00231 Preprocessor1_Model03
##  6 0.00000000101  0.155  roc_auc  binary     0.659     5 0.0103  Preprocessor1_Model03
##  7 0.00292        0.265  accuracy binary     0.962     5 0.00233 Preprocessor1_Model04
##  8 0.00292        0.265  roc_auc  binary     0.666     5 0.0117  Preprocessor1_Model04
##  9 0.0000217      0.296  accuracy binary     0.962     5 0.00231 Preprocessor1_Model05
## 10 0.0000217      0.296  roc_auc  binary     0.659     5 0.0103  Preprocessor1_Model05
## # ... with 20 more rows
## # i Use 'print(n = ...)' to see more rows
```

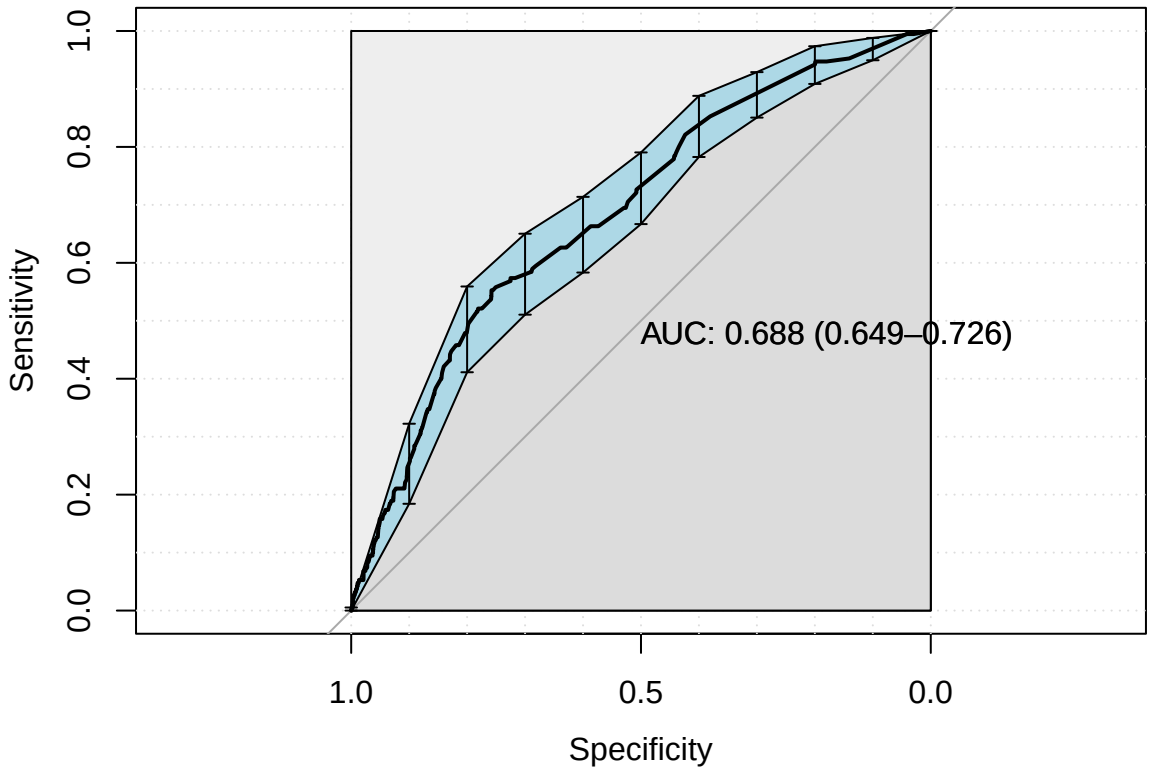
```
autoplot(glmnet_tune, metric = "roc_auc")
```



```
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 0.0337          0.474  roc_auc binary     0.675     5  0.0102 Preprocessor1_Model08
## 2 0.00164         0.750  roc_auc binary     0.668     5  0.0110 Preprocessor1_Model12
## 3 0.00292         0.265  roc_auc binary     0.666     5  0.0117 Preprocessor1_Model04
## 4 0.00000581      0.0629 roc_auc binary     0.659     5  0.0103 Preprocessor1_Model01
## 5 0.00000000101   0.155  roc_auc binary     0.659     5  0.0103 Preprocessor1_Model03
```

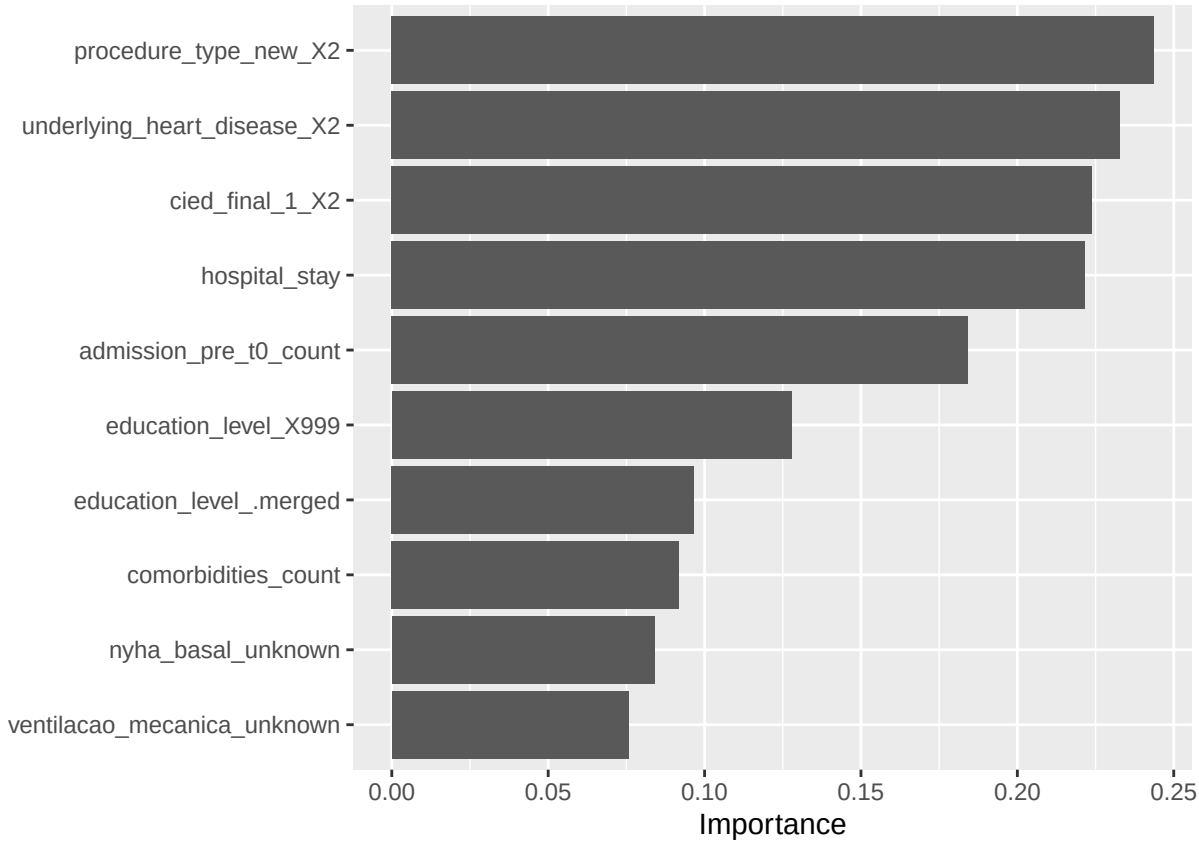
```
best_glmnet <- glmnet_tune %>%
  select_best("roc_auc")

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: 0.04"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3441   85
##    1 1099  105
```

```
##
##                Accuracy : 0.7497
##                  95% CI : (0.7371, 0.762)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0873
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.75793
##             Specificity : 0.55263
##          Pos Pred Value : 0.97589
##          Neg Pred Value : 0.08721
##              Prevalence : 0.95983
##          Detection Rate : 0.72748
##    Detection Prevalence : 0.74545
##       Balanced Accuracy : 0.65528
##
##        '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')
```

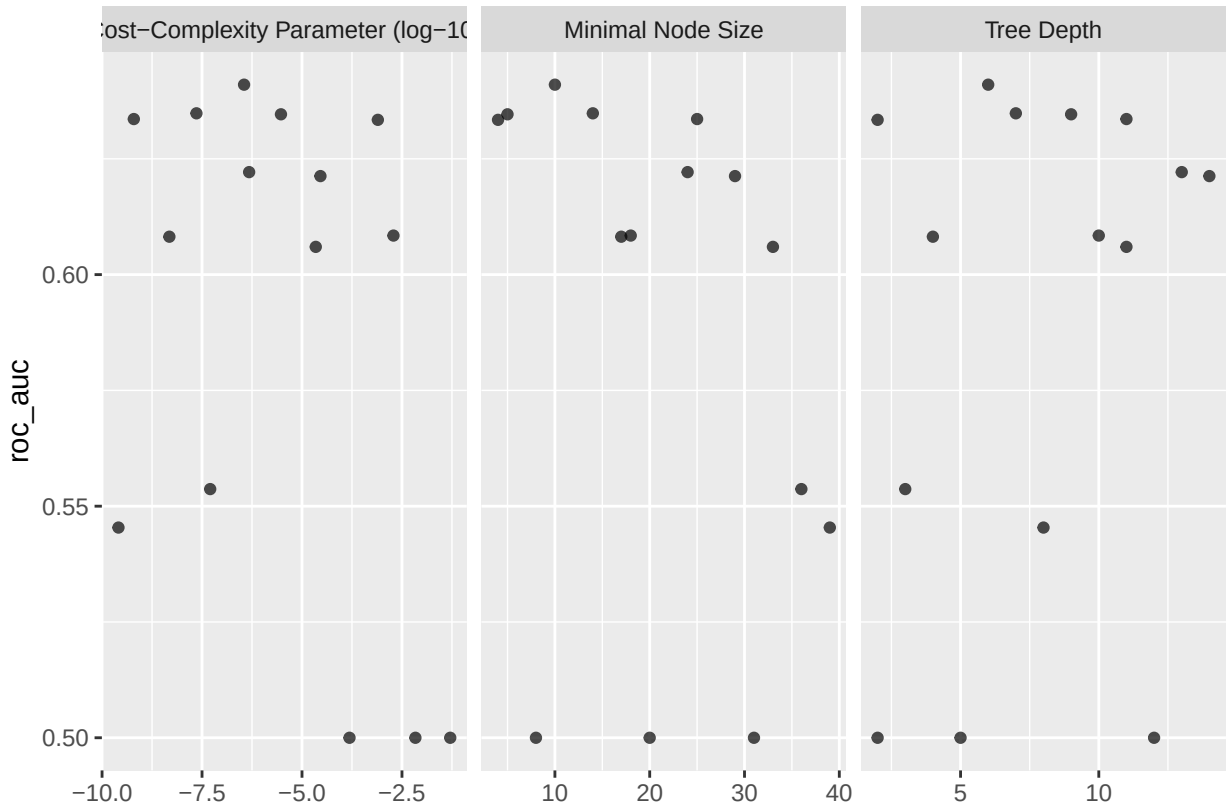


Minutes to run: 0.962

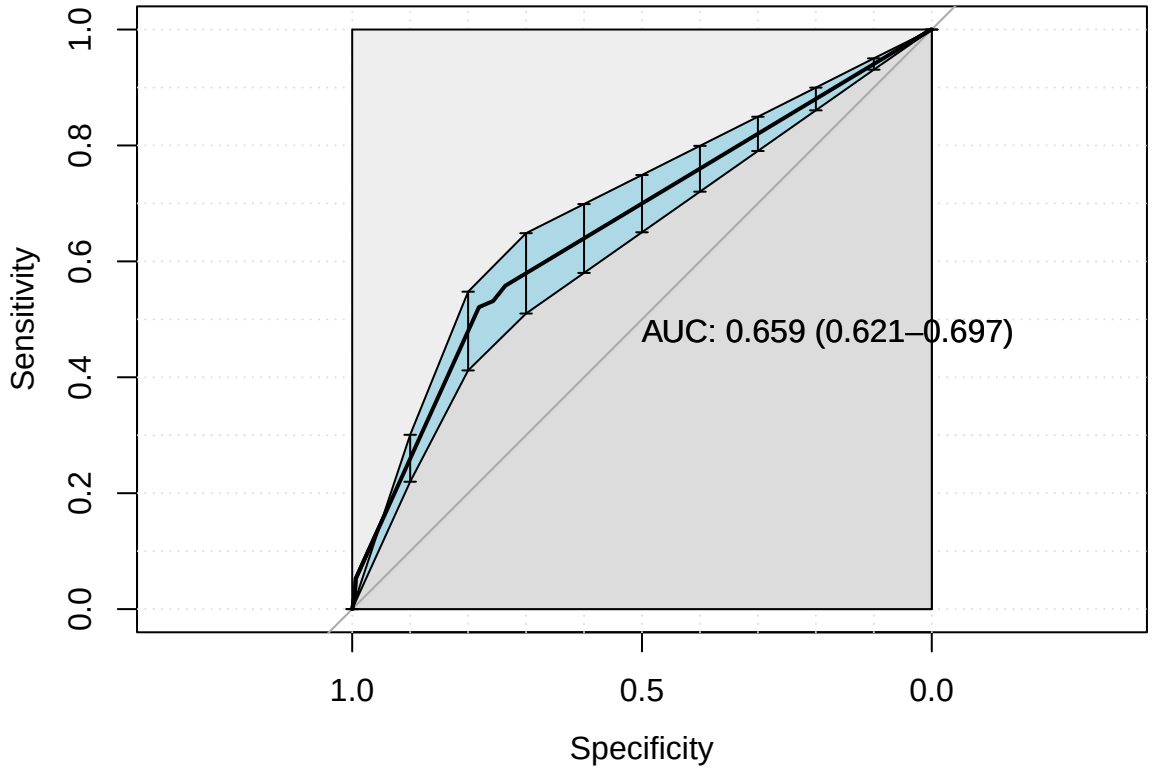
Decision Tree

```
tree_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
```

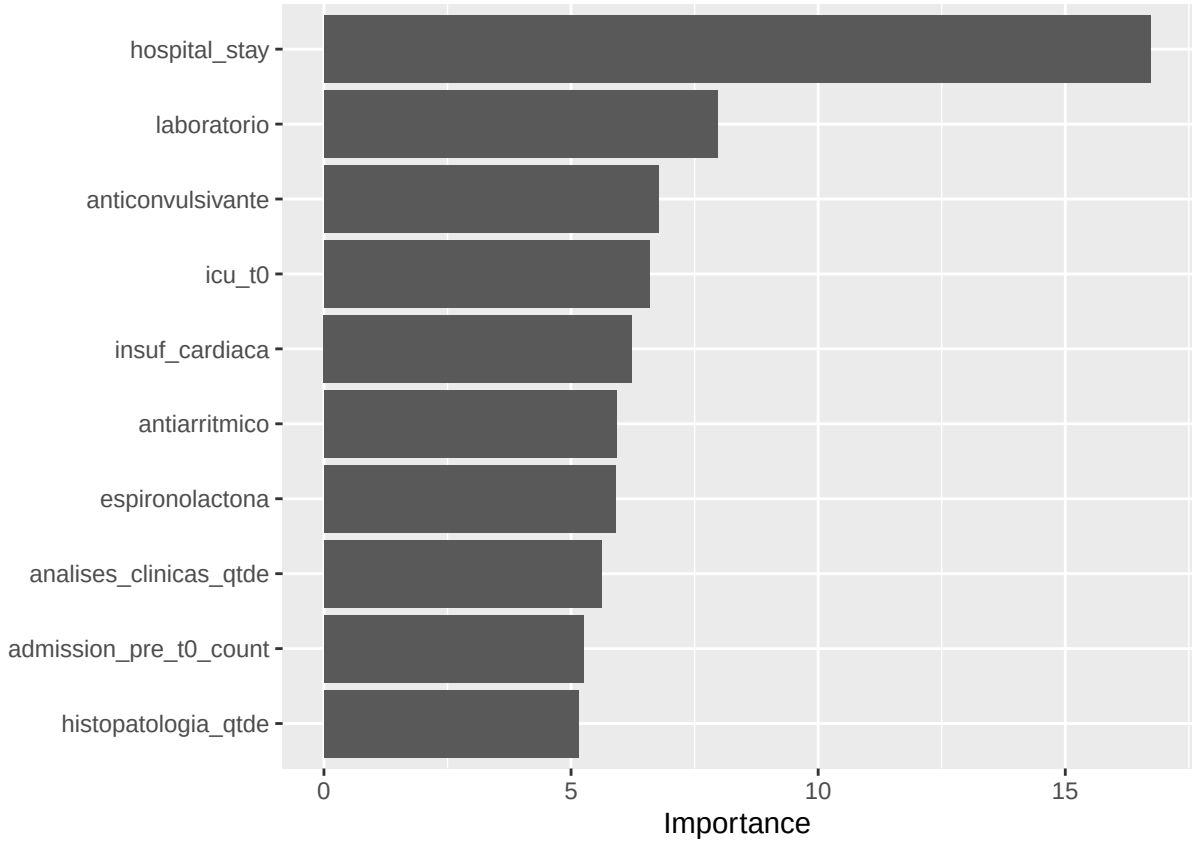
```
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

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")
```



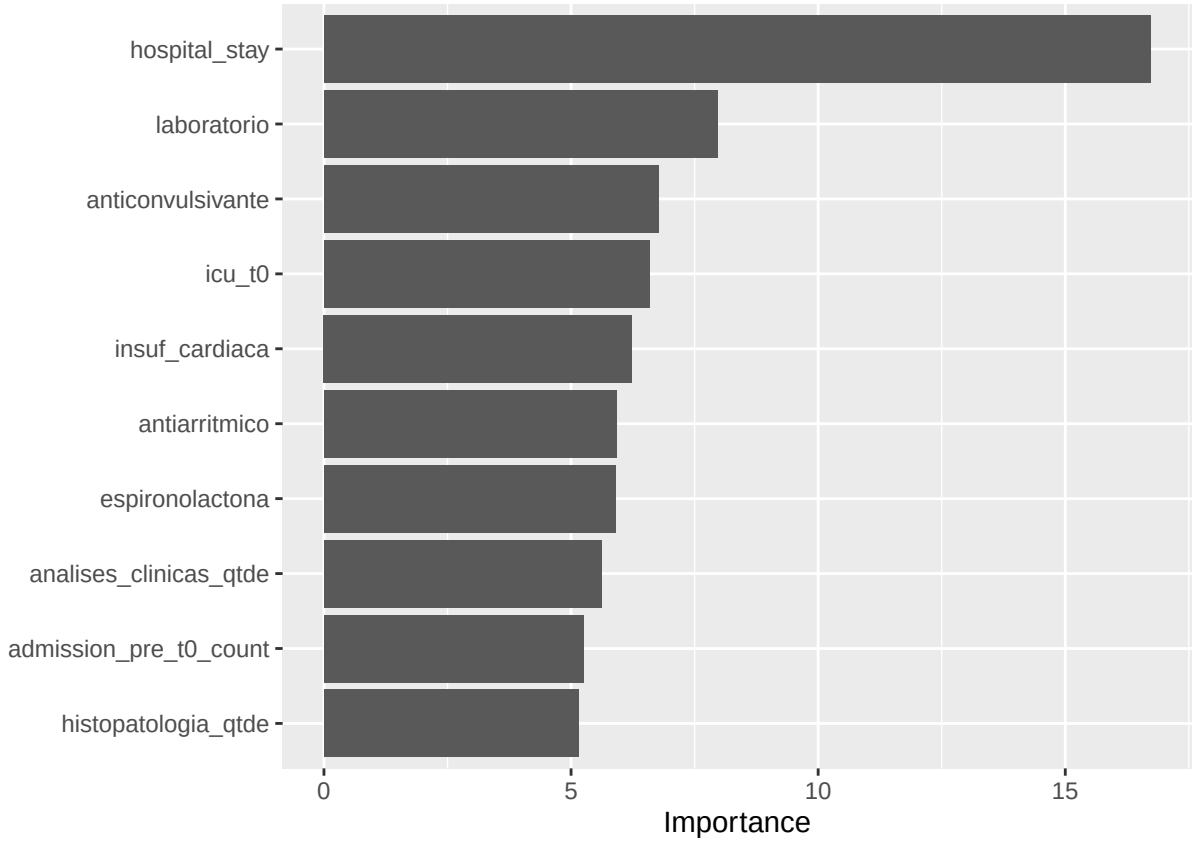
```r
tree_tune %>%
  show_best("roc_auc")

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)
```




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



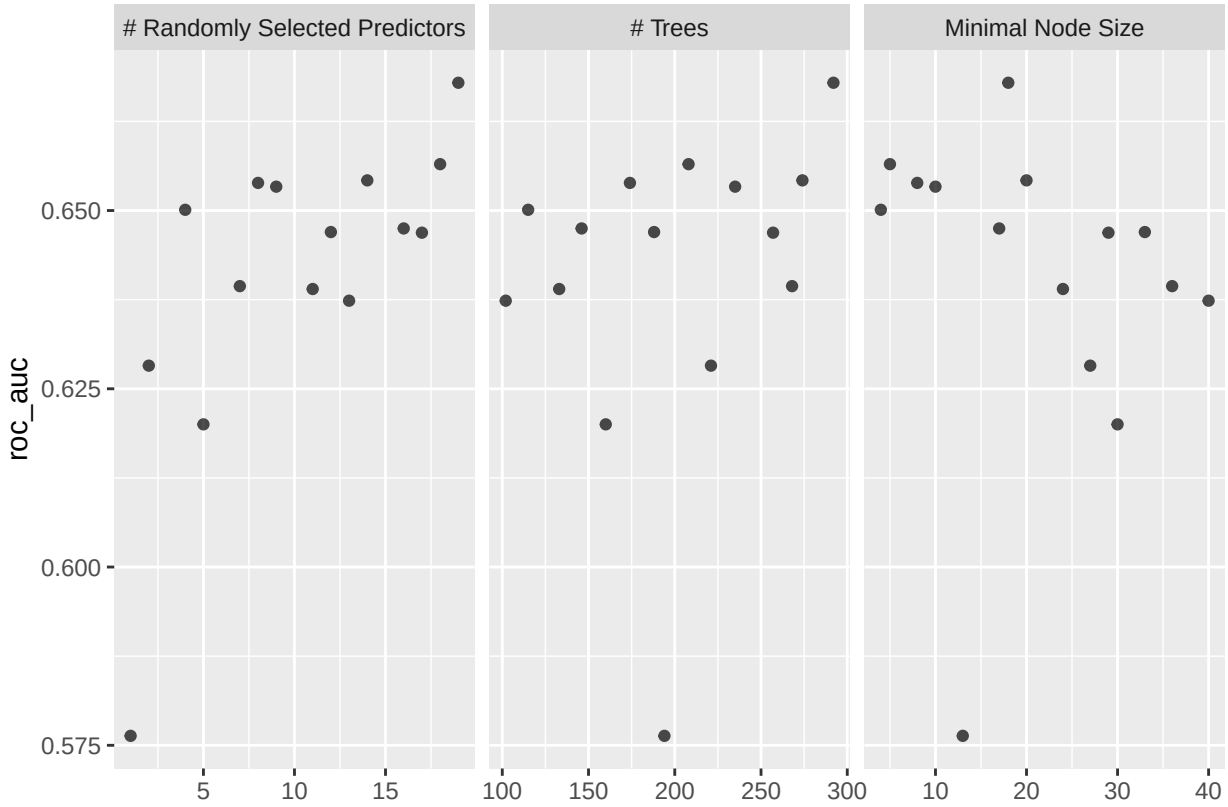

```
extract_vip(final_tree_fit, pred_wrapper = predict,
            reference_class = "0", use_matrix = FALSE,
            method = 'model')
```



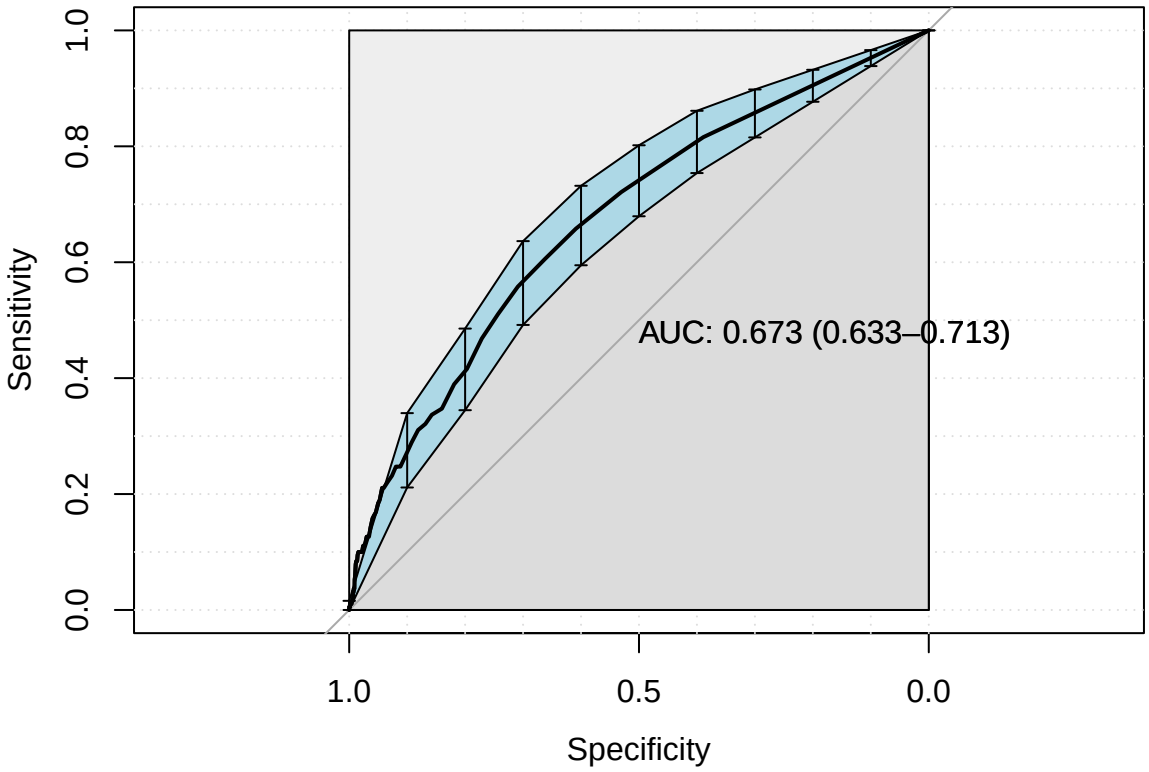
Minutes to run: 1.583

Random Forest

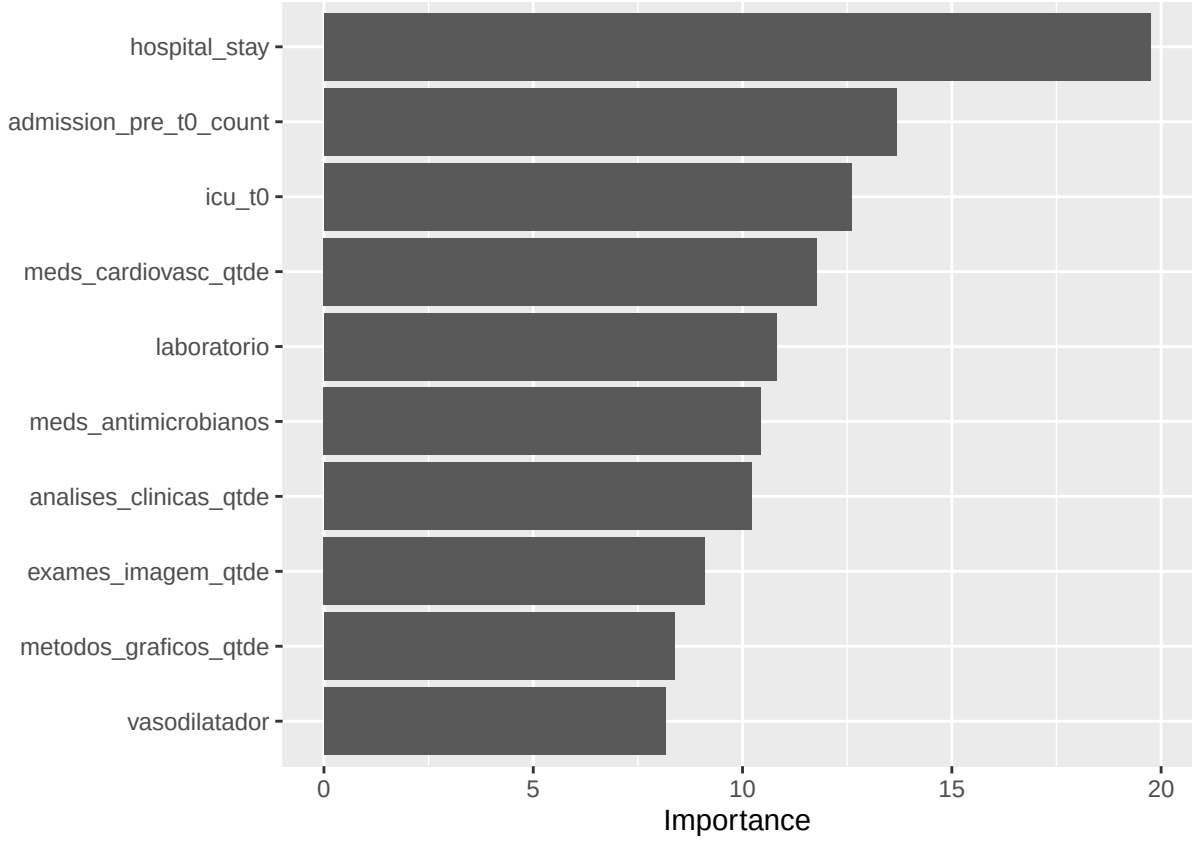
```
rf_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors())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



Minutes to run: 19.824

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

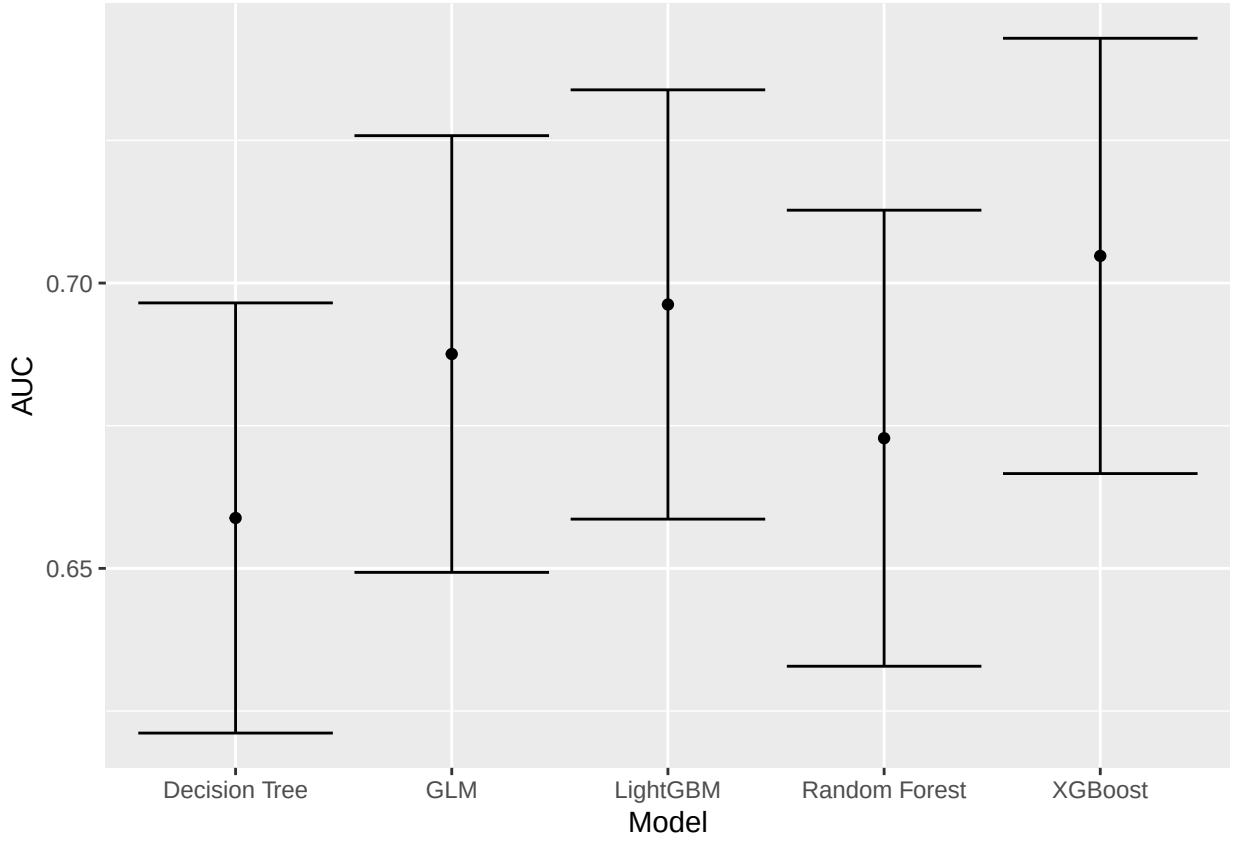
```
# svm_recipe <-
#   recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
#   step_novel(all_nominal_predictors()) %>%
#   step_unknown(all_nominal_predictors()) %>%
#   step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
#   step_dummy(all_nominal_predictors()) %>%
#   step_zv(all_predictors()) %>%
#   step_impute_mean(all_numeric_predictors())
#
# svm_spec <-
#   svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
#   set_mode("classification") %>%
#   set_engine("kernlab")
#
# svm_grid <- grid_latin_hypercube(cost(),
#                                  rbf_sigma(),
#                                  size = grid_size)
#
# svm_workflow <-
#   workflow() %>%
#   add_recipe(svm_recipe) %>%
#   add_model(svm_spec)
#
# svm_tune <-
#   svm_workflow %>%
#   tune_grid(resamples = df_folds,
#             grid = grid_size)
#
# svm_tune %>%
#   collect_metrics()
#
# autoplot(svm_tune, metric = "roc_auc")
#
# svm_tune %>%
#   show_best("roc_auc")
#
# best_svm <- svm_tune %>%
#   select_best("roc_auc")
#
# final_svm_workflow <-
#   svm_workflow %>%
#   finalize_workflow(best_svm)
#
# last_svm_fit <-
#   final_svm_workflow %>%
#   last_fit(df_split)
#
# final_svm_fit <- extract_workflow(last_svm_fit)
#
# svm_auc = validation(final_svm_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