Final Model

Eduardo Yuki Yada

Imports

```
library(tidyverse)
library(yaml)
library(tidymodels)
library(usemodels)
library(vip)

library(SHAPforxgboost)
library(xgboost)
library(Matrix)
library(mltools)
library(bonsai)
library(lightgbm)
```

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
```

Minutes to run: 0.001

Filtering eligible pacients

```
df = df %>%
  filter(disch_outcomes_t0 == 0)

df %>% dim
```

```
## [1] 15766   239
```

Minutes to run: 0.005

Eligible features

```
eligible_columns = df_names %>%
  filter(momento.aquisicao == 'Admissão t0') %>%
  .$variable.name

exception_columns = c('death_intraop', 'death_intraop_1')
```

```r
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
                       )

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. race
## 04. education_level
## 05. patient_state
## 06. underlying_heart_disease
## 07. heart_disease
## 08. nyha_basal
## 09. prior_mi
## 10. heart_failure
## 11. af
## 12. cardiac_arrest
## 13. transplant
## 14. valvopathy
## 15. endocardites
## 16. diabetes
## 17. renal_failure
## 18. hemodialysis
## 19. copd
## 20. comorbidities_count
## 21. procedure_type_1
## 22. reop_type_1
## 23. procedure_type_new
## 24. cied_final_1
## 25. cied_final_group_1
## 26. admission_pre_t0_count
## 27. admission_pre_t0_180d
## 28. year_adm_t0
## 29. icu_t0
## 30. dialysis_t0
## 31. disch_outcomes_t0
## 32. admission_t0_emergency
## 33. aco
## 34. antiarritmico
## 35. betabloqueador
## 36. ieca_bra
## 37. dva
## 38. digoxina
## 39. estatina
## 40. diuretico
## 41. vasodilatador
```

```
## 42. insuf_cardiaca
## 43. espironolactona
## 44. bloq_calcio
## 45. antiplaquetario_ev
## 46. insulina
## 47. anticonvulsivante
## 48. psicofarmacos
## 49. antifungico
## 50. antiviral
## 51. antiretroviral
## 52. classe_meds_qtde
## 53. meds_cardiovasc_qtde
## 54. meds_antimicrobianos
## 55. vni
## 56. cec
## 57. transplante_cardiaco
## 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. ecg
## 71. holter
## 72. teste_esforco
## 73. espiro_ergoespiro
## 74. tilt_teste
## 75. metodos_graficos_qtde
## 76. laboratorio
## 77. cultura
## 78. analises_clinicas_qtde
## 79. citologia
## 80. biopsia
## 81. histopatologia_qtde
## 82. angio_rm
## 83. angio_tc
## 84. arteriografia
## 85. cintilografia
## 86. ecocardiograma
## 87. endoscopia
## 88. flebografia
## 89. pet_ct
## 90. ultrassom
## 91. tomografia
## 92. radiografia
## 93. ressonancia
## 94. exames_imagem_qtde
## 95. bic
## 96. mpp
```

Minutes to run: 0

Train test split (70%/30%)

```
set.seed(42)

df[columns_list$outcome_columns] <- lapply(df[columns_list$outcome_columns], factor)
df <- mutate(df, across(where(is.character), as.factor))

df_split <- initial_split(df %>% dplyr::select(all_of(c(features, outcome_column))),
                          prop = .7, strata = all_of(outcome_column))
df_train <- training(df_split)
df_test <- testing(df_split)

dim(df_train)[1] / dim(df)[1]
```

```
## [1] 0.6999239
```

```
dim(df_test)[1] / dim(df)[1]
```

```
## [1] 0.3000761
```

Minutes to run: 0.003

Global parameters

```
k <- 4 # Number of folds for cross validation
grid_size <- 50 # Number of parameter combination to tune on each model

set.seed(234)
df_folds <- vfold_cv(df_train, v = k,
                     strata = all_of(outcome_column))

max_auc_loss <- 0.01
```

Minutes to run: 0

Functions

```
validation = function(model_fit, new_data, plot=TRUE) {
  library(pROC)
  library(caret)

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

  pROC_obj <- roc(
    new_data[[outcome_column]],
    test_predictions_prob,
    direction = "<",
    levels = c(0, 1),
    smoothed = TRUE,
    ci = TRUE,
    ci.alpha = 0.9,
    stratified = FALSE,
    plot = plot,
    auc.polygon = TRUE,
    max.auc.polygon = TRUE,
    grid = TRUE,
    print.auc = TRUE,
    show.thres = TRUE
```

```
  )

  test_predictions_class <-
    predict(model_fit, new_data = new_data, type = "class") %>%
    rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
    .$class

  conf_matrix <- table(test_predictions_class, new_data[[outcome_column]])

  if (plot) {
    sens.ci <- ci.se(pROC_obj)
    plot(sens.ci, type = "shape", col = "lightblue")
    plot(sens.ci, type = "bars")

    confusionMatrix(conf_matrix) %>% print
  }

  return(pROC_obj)
}
```

Minutes to run: 0

Feature Selection

```
model_fit_wf <- function(features, outcome_column, hyperparameters){
  model_recipe <-
    recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
           data = df_train %>% select(all_of(c(features, outcome_column)))) %>%
    step_novel(all_nominal_predictors()) %>%
    step_unknown(all_nominal_predictors()) %>%
    step_other(all_nominal_predictors(), threshold = 0.05, other=".merged") %>%
    step_impute_mean(all_numeric_predictors()) %>%
    step_zv(all_predictors())

  model_spec <-
    do.call(boost_tree, hyperparameters) %>%
    set_engine("lightgbm") %>%
    set_mode("classification")

  model_workflow <-
    workflow() %>%
    add_recipe(model_recipe) %>%
    add_model(model_spec)

  model_fit_rs <- model_workflow %>%
    fit_resamples(df_folds)

  model_fit <- model_workflow %>%
    fit(df_train)

  model_auc <- validation(model_fit, df_test, plot=F)

  raw_model <- parsnip::extract_fit_engine(model_fit)

  feature_importance <- lgb.importance(raw_model, percentage = TRUE)

  return(list(cv_auc = collect_metrics(model_fit_rs) %>% filter(.metric == 'roc_auc') %>% .$mean,
              importance = feature_importance,
              auc = as.numeric(model_auc$auc),
              auc_lower = model_auc$ci[1],
```

```
              auc_upper = model_auc$ci[3]))
}
```

Minutes to run: 0

```
hyperparameters <- readRDS(
  sprintf(
    "../EDA/auxiliar/hyperparameters/model_selection/lightgbm_parameters_%s.rds",
    outcome_column
  )
)

full_model <- model_fit_wf(features, outcome_column, hyperparameters)

sprintf('Full Model CV Train AUC: %.3f' ,full_model$cv_auc)
```

```
## [1] "Full Model CV Train AUC: 0.711"
```

```
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)
```

```
## [1] "Full Model Test AUC: 0.702"
```

Minutes to run: 0.138

Features with zero importance on the initial model:

```
unimportant_features <- setdiff(features, full_model$importance$Feature)

unimportant_features %>%
  gluedown::md_order()
```

```
## 01. transplant
## 02. endocardites
## 03. hemodialysis
## 04. dialysis_t0
## 05. disch_outcomes_t0
## 06. antiplaquetario_ev
## 07. antiviral
## 08. antiretroviral
## 09. transplante_cardiaco
## 10. icp
## 11. intervencao_cv
## 12. angioplastia
## 13. transfusao
## 14. tilt_teste
## 15. citologia
## 16. angio_rm
## 17. arteriografia
## 18. pet_ct
```

Minutes to run: 0

```
trimmed_features <- full_model$importance$Feature
hyperparameters$mtry = min(hyperparameters$mtry, length(trimmed_features))
trimmed_model <- model_fit_wf(trimmed_features,
                              outcome_column, hyperparameters)

sprintf('Trimmed Model CV Train AUC: %.3f' ,trimmed_model$cv_auc)
```

```
## [1] "Trimmed Model CV Train AUC: 0.712"
```

```
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)
```

```
## [1] "Trimmed Model Test AUC: 0.707"
```

Minutes to run: 0.128

```
current_features <- trimmed_features
current_model <- trimmed_model
current_least_important <- tail(trimmed_model$importance$Feature, 1)
current_auc_loss <- full_model$cv_auc - trimmed_model$cv_auc

selection_results <- tibble::tribble(
  ~`Number of Features`, ~`AUC Loss`, ~`Least Important Feature`,
  length(features), 0, tail(full_model$importance$Feature, 1),
  length(trimmed_features), current_auc_loss, tail(trimmed_model$importance$Feature, 1)
)

while (current_auc_loss < max_auc_loss){
  last_feature_droped <- current_least_important

  current_features <- setdiff(current_features, current_least_important)
  hyperparameters$mtry = min(hyperparameters$mtry, length(current_features))
  current_model <- model_fit_wf(current_features, outcome_column, hyperparameters)
  current_least_important <- tail(current_model$importance$Feature, 1)

  current_auc_loss <- full_model$cv_auc - current_model$cv_auc

  selection_results <- selection_results %>%
    add_row(`Number of Features` = length(current_features),
            `AUC Loss` = current_auc_loss,
            `Least Important Feature` = current_least_important)

  print(c(length(current_features), current_auc_loss))
}
```

```
## [1] 77.000000000 -0.001917419
## [1] 76.000000000 -0.001253666
## [1] 75.0000000000 -0.0009763942
## [1] 74.000000000 -0.001101279
## [1] 73.0000000000 -0.0008724818
## [1] 72.00000000 -0.00146388
## [1] 71.000000000 -0.001525097
## [1] 70.000000000 -0.001362892
## [1] 69.000000000 -0.001624644
## [1] 68.000000000 -0.001816426
## [1] 67.0000000000 -0.0008624337
## [1] 66.000000000 -0.001092141
## [1] 65.000000000 -0.001699777
## [1] 64.0000000000 -0.0008455681
## [1] 63.000000000 -0.000996844
## [1] 62.00000000 -0.00119166
## [1] 61.000000000 -0.001258677
## [1] 6.000000e+01 6.473615e-05
## [1] 59.0000000000 -0.0003934806
## [1] 58.0000000000 -0.0008327022
## [1] 57.000000000 -0.001321629
## [1] 56.000000000 -0.001005616
## [1] 55.0000000000 -0.0007493073
## [1] 54.000000000 -0.001237998
## [1] 5.300000e+01 2.831278e-04
## [1] 5.200000e+01 1.067679e-04
## [1] 5.100000e+01 8.548722e-04
## [1] 50.000000000  0.001263115
## [1] 49.000000000  0.001462437
## [1] 4.800000e+01 5.791697e-04
## [1] 47.000000000  0.001213272
## [1] 46.000000000  0.001340302
```

```
## [1] 45.000000000  0.001195663
## [1] 44.000000000  0.001070179
## [1] 43.000000000  0.001790489
## [1] 42.000000000  0.002324188
## [1] 41.000000000  0.002109332
## [1] 40.000000000  0.003613047
## [1] 39.000000000  0.003339984
## [1] 38.000000000  0.003829213
## [1] 37.000000000  0.005592671
## [1] 36.000000000  0.006868831
## [1] 35.000000000  0.007959311
## [1] 34.000000000  0.008344494
## [1] 33.00000000  0.01116052
```

```
selection_results
```

```
## # A tibble: 47 x 3
##    `Number of Features` `AUC Loss` `Least Important Feature`
##                   <int>      <dbl> <chr>
##  1                   96   0        vni
##  2                   78  -0.00147  heart_disease
##  3                   77  -0.00192  cve_desf
##  4                   76  -0.00125  flebografia
##  5                   75  -0.000976 mpp
##  6                   74  -0.00110  cec
##  7                   73  -0.000872 copd
##  8                   72  -0.00146  endoscopia
##  9                   71  -0.00153  teste_esforco
## 10                   70  -0.00136  renal_failure
## # ... with 37 more rows
```

Minutes to run: 5.309

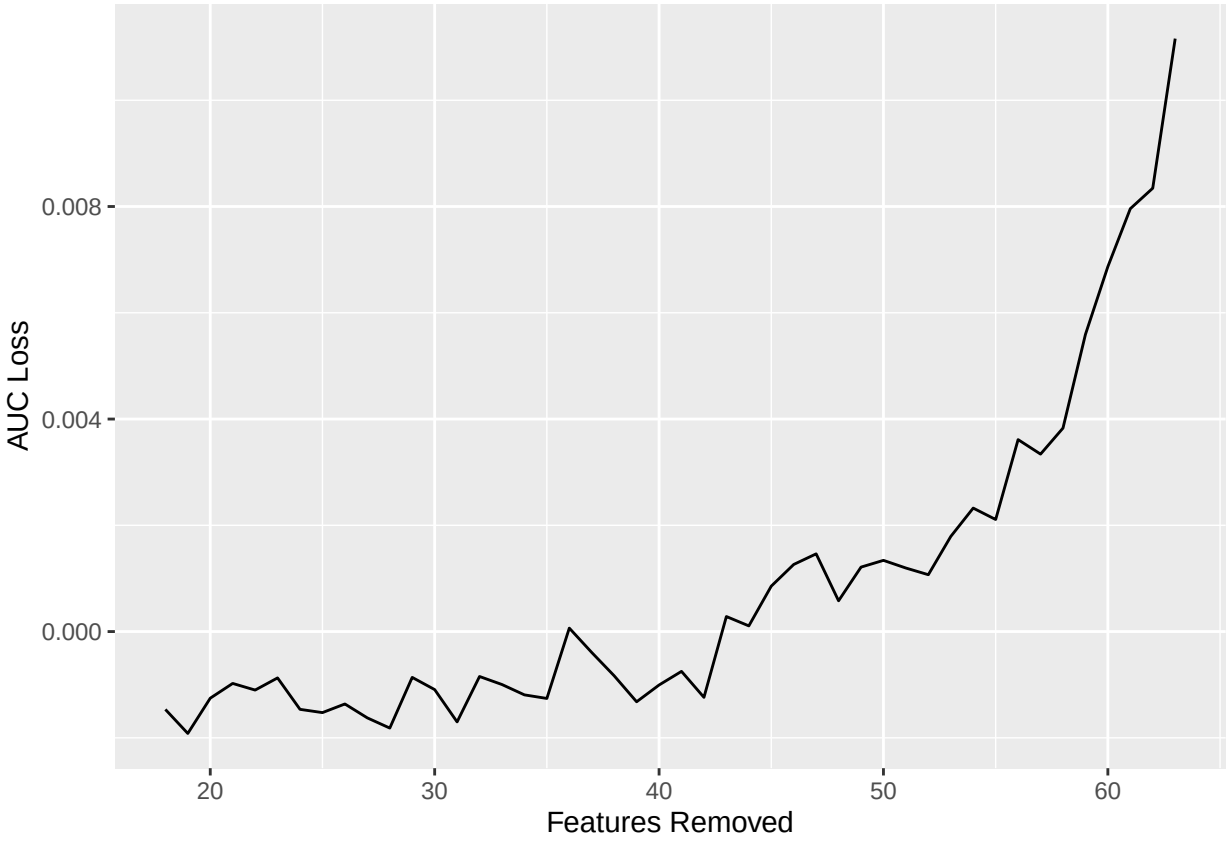
```
selected_features <- c(current_features, last_feature_droped)

feature_selected_model <- model_fit_wf(selected_features,
                                       outcome_column, hyperparameters)

sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)
```

```
## [1] "Trimmed Model CV Train AUC: 0.702"
```

```
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)
```

```
## [1] "Trimmed Model Test AUC: 0.695"
```

```
selection_results %>%
  filter(`Number of Features` < length(features)) %>%
  mutate(`Features Removed` = length(features) - `Number of Features`) %>%
  ggplot(aes(x = `Features Removed`, y = `AUC Loss`)) +
  geom_line()
```



Minutes to run: 0.099

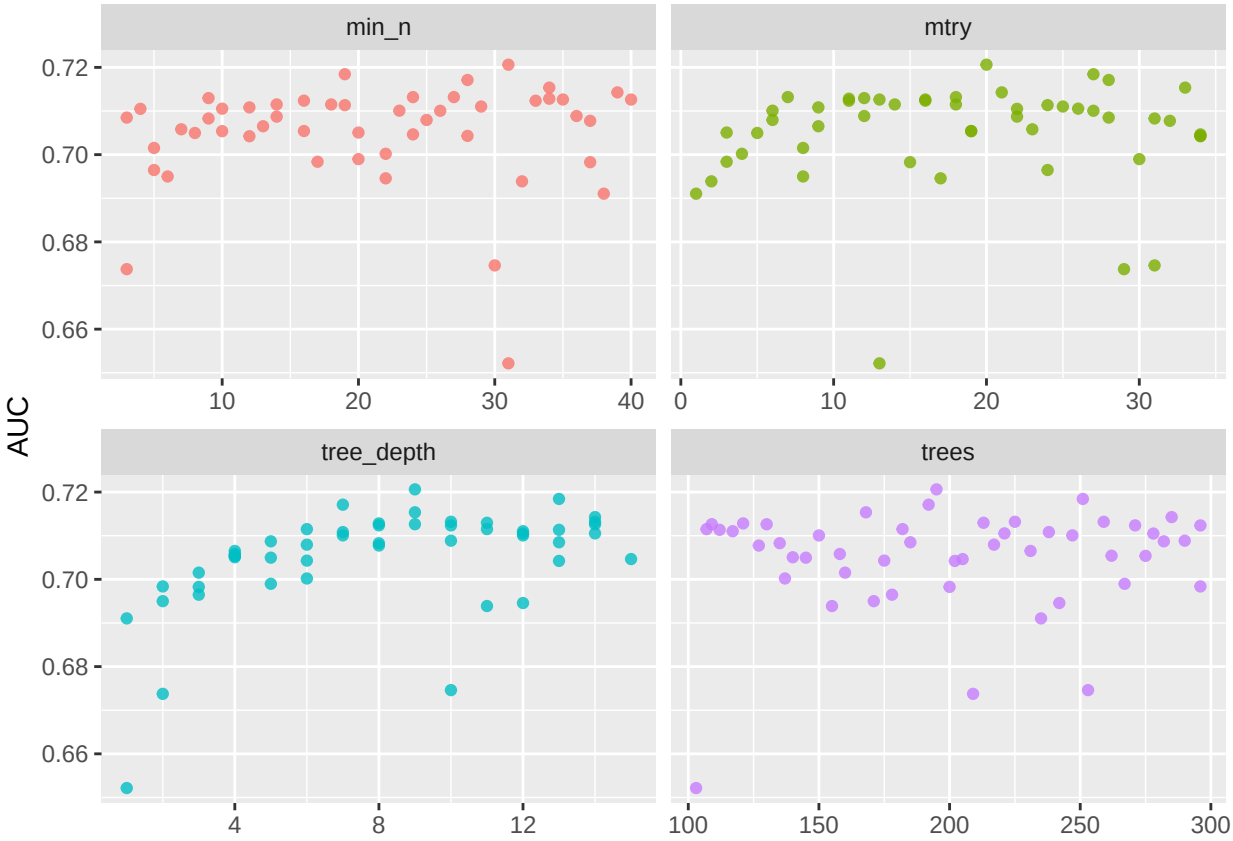
Hyperparameter tuning

```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% dplyr::select(all_of(c(selected_features, outcome_column)))) %>%
  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(), one_hot = TRUE) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_zv(all_predictors())

lightgbm_spec <- boost_tree(
  mtry = tune(),
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  loss_reduction = tune()
) %>%
  set_engine("lightgbm") %>%
  set_mode("classification")

lightgbm_grid <- grid_latin_hypercube(
  finalize(mtry(),
           df_train %>% dplyr::select(all_of(c(selected_features, outcome_column)))),
  dials::trees(range = c(100L, 300L)),
  min_n(),
  tree_depth(),
  learn_rate(),
  loss_reduction(),
  size = grid_size
)
```

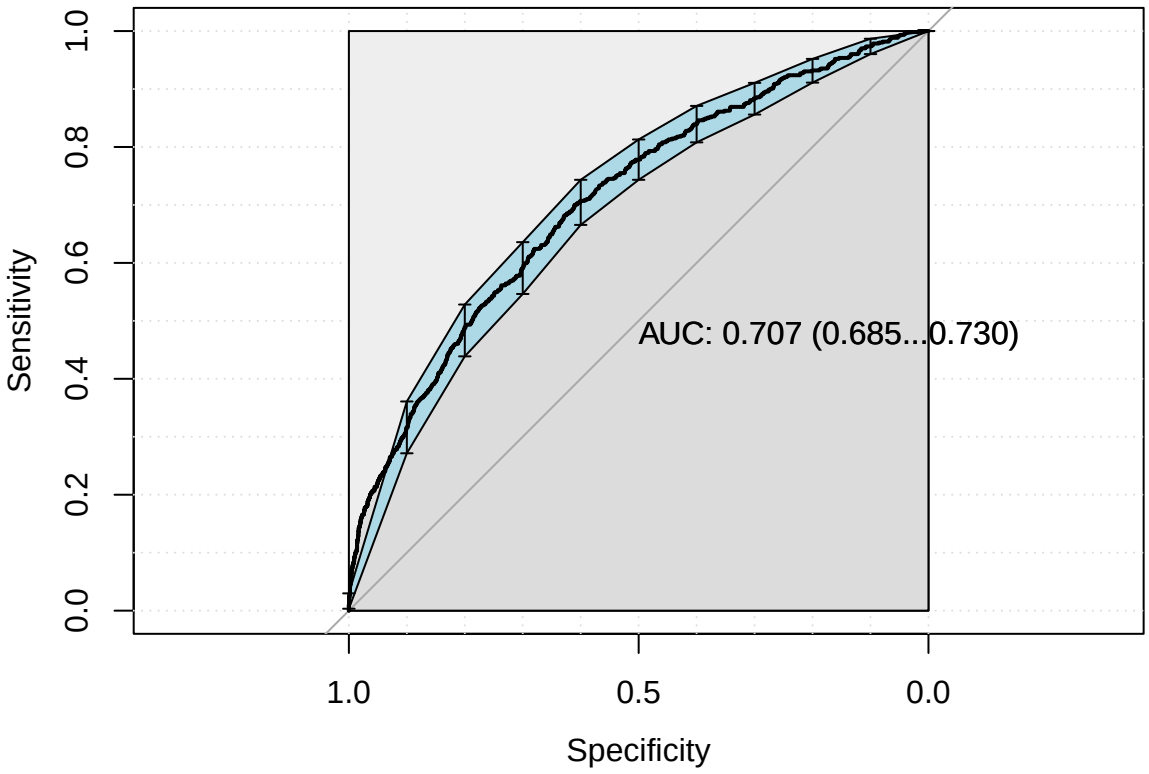
```
lightgbm_workflow <-
  workflow() %>%
  add_recipe(lightgbm_recipe) %>%
  add_model(lightgbm_spec)

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 .estimator  mean     n std_err .config
##   <int> <int> <int>      <int>      <dbl>          <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    20   195    31          9   0.0171       0.996      roc_auc binary     0.721     4 0.00232 Preprocessor1
## 2    27   251    19         13   0.0213       0.00000377 roc_auc binary     0.718     4 0.00331 Preprocessor1
## 3    28   192    28          7   0.00675      0.0980     roc_auc binary     0.717     4 0.00426 Preprocessor1
## 4    33   168    34          9   0.0407       0.00105    roc_auc binary     0.715     4 0.00330 Preprocessor1
## 5    21   285    39         14   0.000920     0.0000149  roc_auc binary     0.714     4 0.00513 Preprocessor1
```

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

lightgbm_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, mtry:tree_depth) %>%
  pivot_longer(mtry:tree_depth,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "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)
```



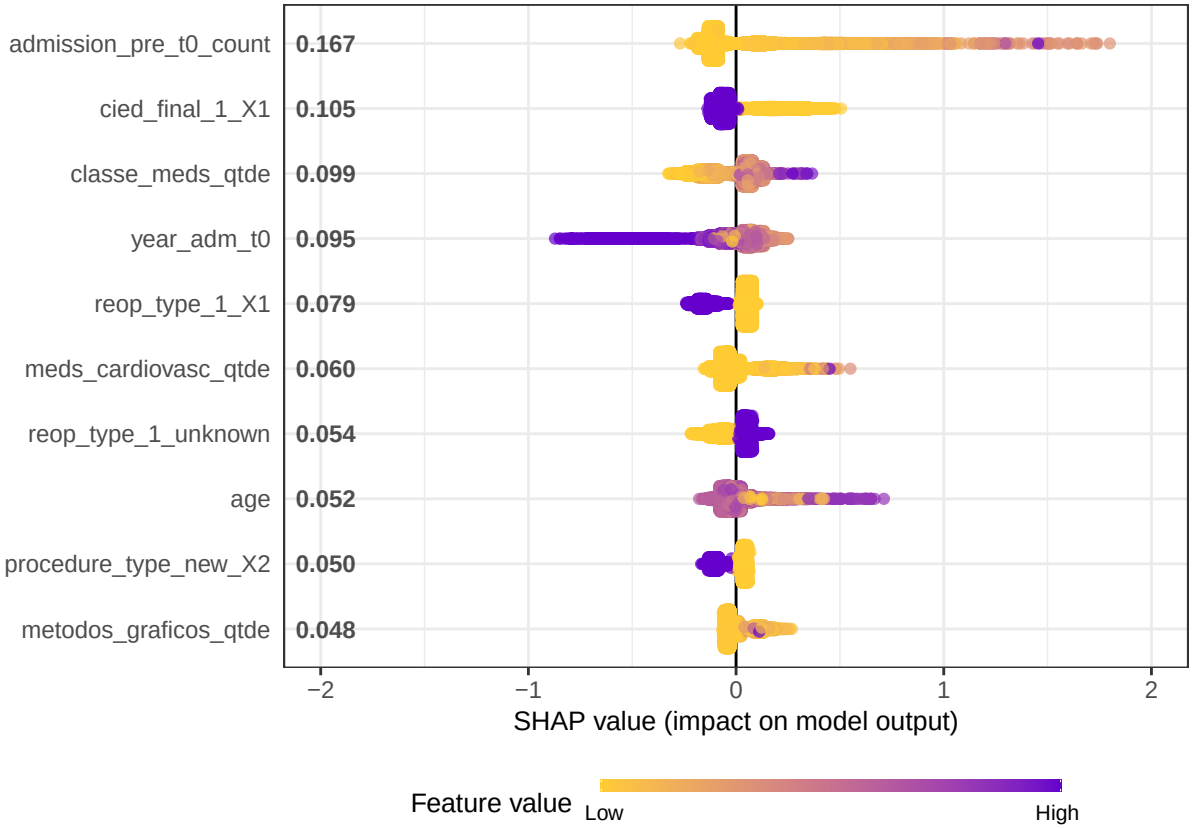

```
##   |
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                      0 4113  574
##                      1   14   30
##
##                Accuracy : 0.8757
##                  95% CI : (0.866, 0.885)
##     No Information Rate : 0.8723
##     P-Value [Acc > NIR] : 0.2507
##
##                   Kappa : 0.0766
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99661
##             Specificity : 0.04967
##          Pos Pred Value : 0.87753
##          Neg Pred Value : 0.68182
##              Prevalence : 0.87233
##          Detection Rate : 0.86937
##    Detection Prevalence : 0.99070
##       Balanced Accuracy : 0.52314
##
##        'Positive' Class : 0
##
```

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

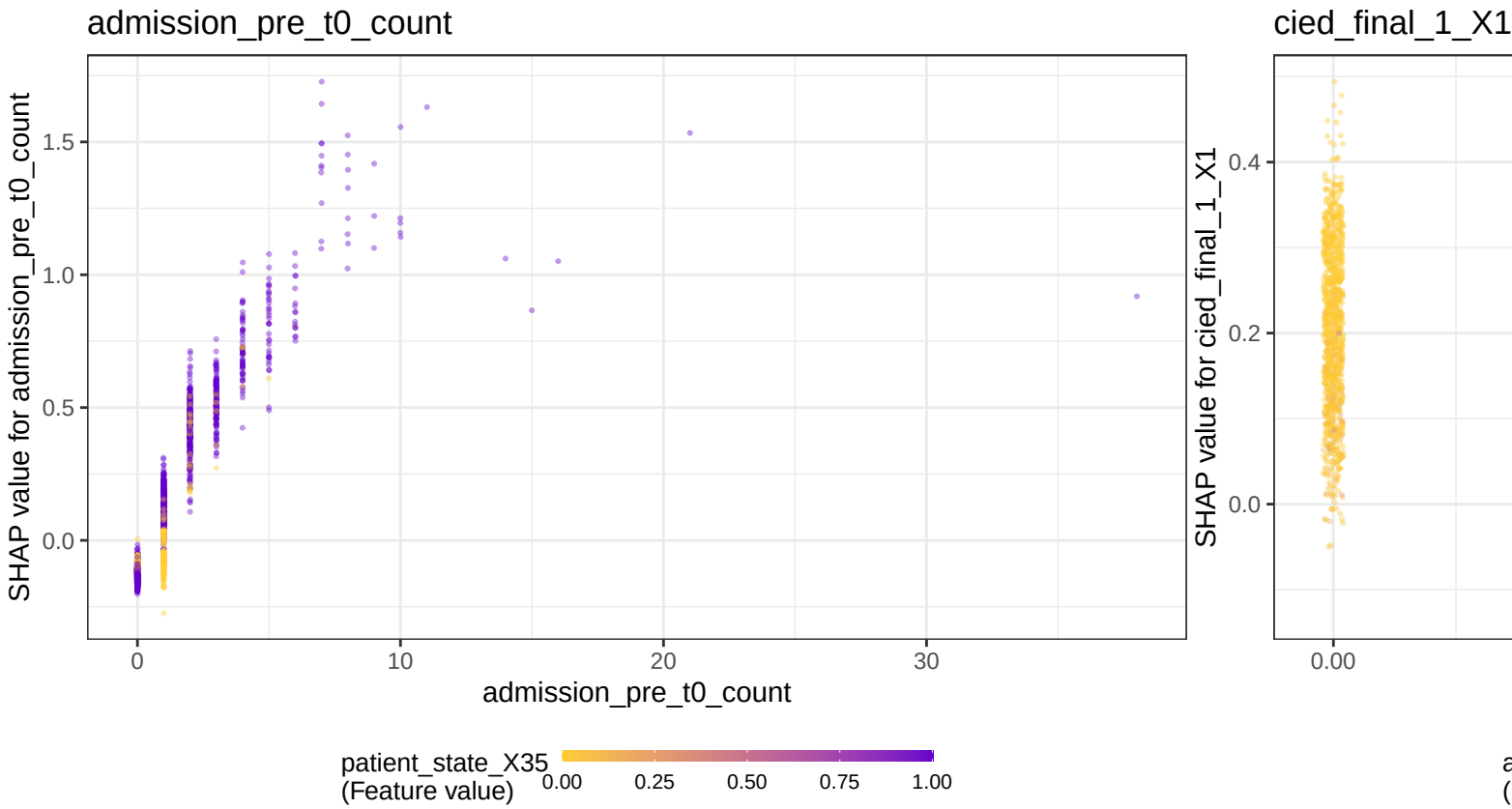
Minutes to run: 1.855

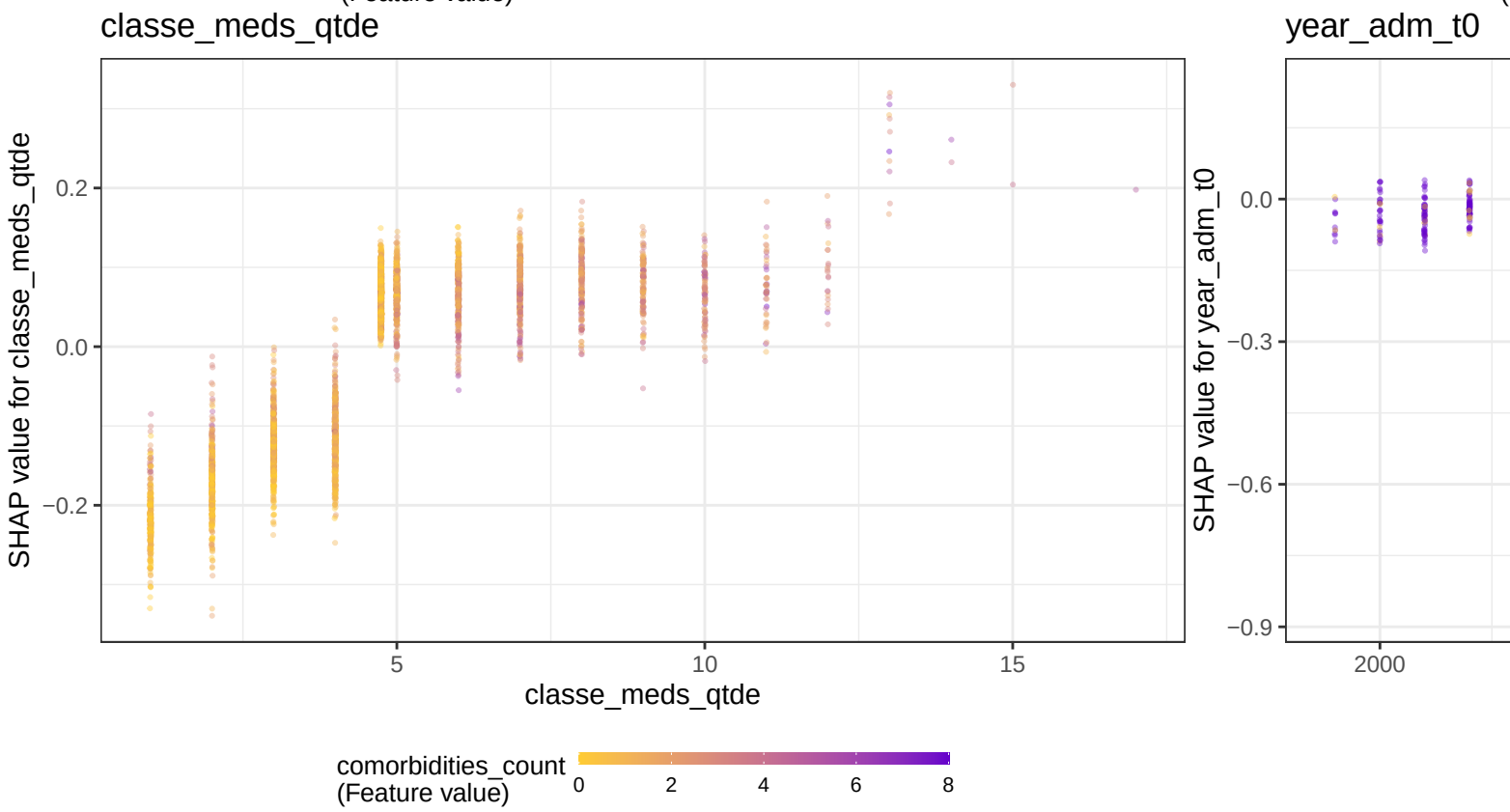
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)
```

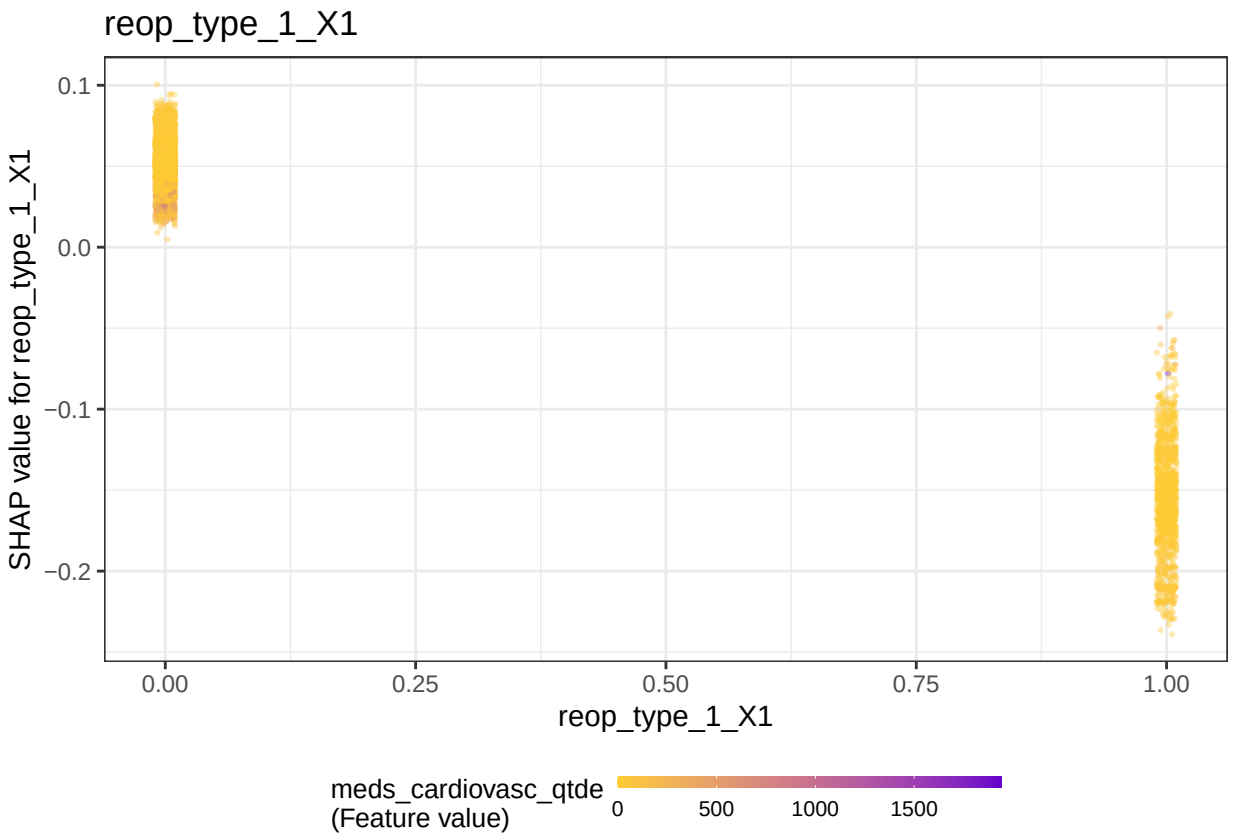
```r
trained_rec <- prep(lightgbm_recipe, training = df_train)

df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

matrix_train <- as.matrix(df_train_baked %>% select(-all_of(outcome_column)))
matrix_test <- as.matrix(df_test_baked %>% select(-all_of(outcome_column)))

shap.plot.summary.wrap1(model = lightgbm_model, X = matrix_train, top_n = 10, dilute = F)
```




```r
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

for (x in shap.importance(shap, names_only = TRUE)[1:5]) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = FALSE,
    jitter_width = 0.01,
    alpha = 0.4
    ) +
  ggtitle(x)
  print(p)
}
```


admission_pre_t0_count
SHAP value for admission_pre_t0_count
admission_pre_t0_count
cied_final_1_X1
SHAP value for cied_final_1_X1
patient_state_X35
(Feature value)
0.00
0.25
0.50
0.75
1.00


classe_meds_qtde
SHAP value for classe_meds_qtde
classe_meds_qtde
year_adm_t0
SHAP value for year_adm_t0
comorbidities_count
(Feature value)
0
2
4
6
8

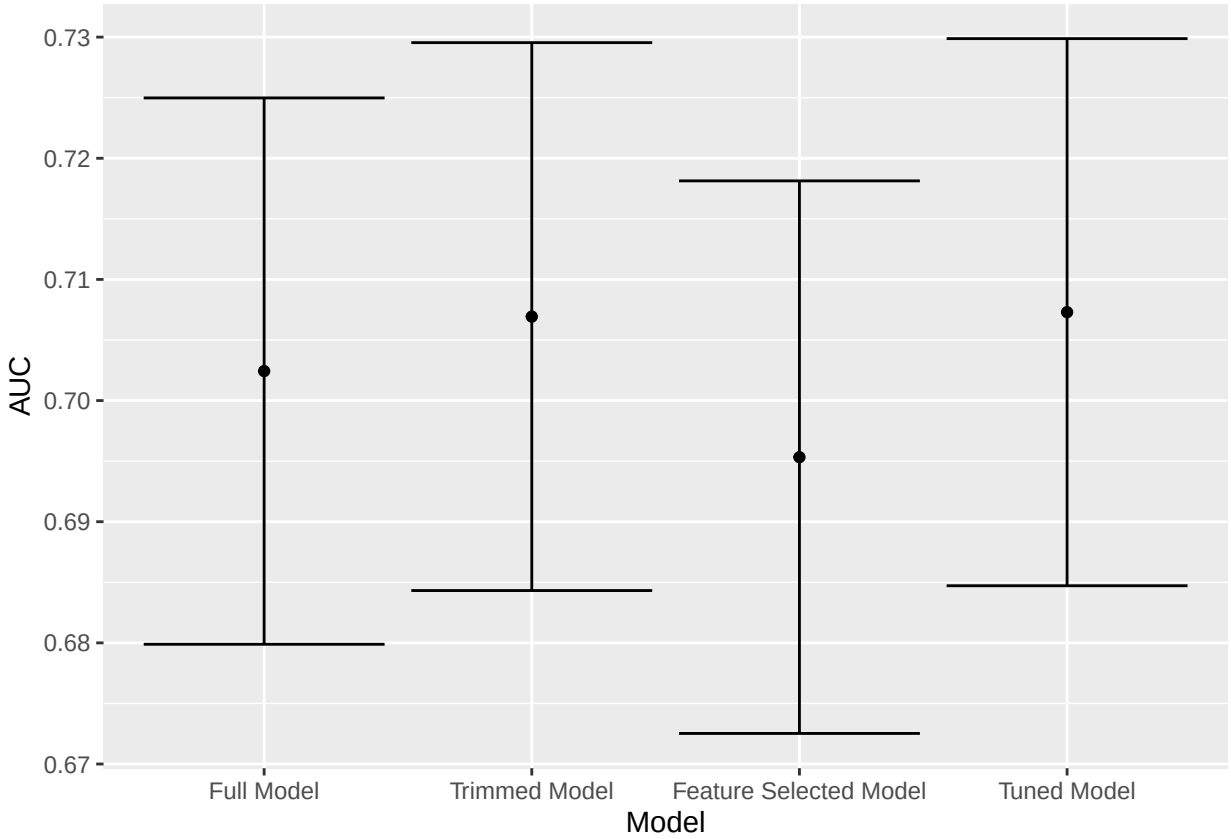


Minutes to run: 0.159

Minutes to run: 0

Models Comparison

```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper,
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper,
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower, feature_selected_model
  'Tuned Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3]
) %>%
  mutate(Target = outcome_column,
         Model = factor(Model,
                        levels = c('Full Model', 'Trimmed Model',
                                   'Feature Selected Model', 'Tuned Model')))

df_auc %>%
  ggplot(aes(x = Model, y = AUC, ymin = `Lower Limit`, ymax = `Upper Limit`)) +
    geom_point() +
    geom_errorbar()
```




```
saveRDS(df_auc, sprintf("../EDA/auxiliar/performance/tuning/%s_auc_result.RData", outcome_column))

# Save the final model
saveRDS(final_lightgbm_fit, sprintf("../EDA/results/%s/final_model.RData", outcome_column))
```

Minutes to run: 0.012