

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: 0.006

Eligible features

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

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

```

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. 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. disch_outcomes_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. cec
## 55. transplante_cardiaco
## 56. outros_proced_cirurgicos
## 57. icp
## 58. intervencao_cv
## 59. angioplastia
## 60. cateterismo
## 61. eletrofisiologia
## 62. cateter_venoso_central
## 63. proced_invasivos_qtde
## 64. cve_desf
## 65. transfusao
## 66. interconsulta
## 67. equipe_multiprof
## 68. ecg
## 69. holter
## 70. teste_esforco
## 71. espiro_ergoespiro
## 72. tilt_teste
## 73. metodos_graficos_qtde
## 74. laboratorio
## 75. cultura
## 76. analises_clinicas_qtde
## 77. citologia
## 78. biopsia
## 79. histopatologia_qtde
## 80. angio_rm
## 81. angio_tc
## 82. cintilografia
## 83. ecocardiograma
## 84. endoscopia
## 85. flebografia
## 86. pet_ct
## 87. ultrassom
## 88. tomografia
## 89. radiografia
## 90. ressonancia
## 91. exames_imagem_qtde
## 92. bic
## 93. 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] 0.6999873  

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

## [1] 0.3000127  

Minutes to run: 0.004

```

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: 0.709"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.690"

Minutes to run: 0.113

Features with zero importance on the initial model:

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

unimportant_features %>%
  gluedown::md_order()

## 01. heart_disease
## 02. transplant
## 03. endocardites
## 04. diabetes
## 05. renal_failure
## 06. hemodialysis
## 07. copd
## 08. dialysis_t0
## 09. disch_outcomes_t0
## 10. antiretroviral
## 11. vni
## 12. cec
## 13. intervencao_cv
## 14. angioplastia
## 15. cve_desf
## 16. tilt_teste
## 17. angio_rm
## 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: 0.708"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

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

Minutes to run: 0.103

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_dropped <- 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] 74.000000000 0.001245168
## [1] 73.000000000 -0.0006558137
## [1] 7.200000e+01 6.549829e-04
## [1] 71.0000000 0.00139658
## [1] 70.000000000 -0.005523003
## [1] 6.900000e+01 4.027343e-05
## [1] 68.000000000 0.001417258
## [1] 67.000000000 -0.000223407
## [1] 66.000000000 -0.001894321
## [1] 65.000000000 -0.0008551798
## [1] 64.000000000 -0.001139875
## [1] 63.000000000 -0.002050912
## [1] 62.000000000 -0.002388728
## [1] 61.000000000 -0.002859197
## [1] 60.000000000 -0.00152455
## [1] 59.000000000 -0.001393156
## [1] 58.000000000 -0.002589035
## [1] 57.000000000 -0.001937365
## [1] 56.000000000 -0.001851293
## [1] 55.000000000 -0.002434957
## [1] 54.000000000 -0.002470455
## [1] 53.000000000 -0.002422712
## [1] 52.000000000 -0.001164484
## [1] 51.000000000 -0.002339423
## [1] 50.000000000 -0.002034353
## [1] 49.000000000 -0.001229278
## [1] 48.000000000 -0.0008798524
## [1] 47.000000000 -0.0008300081
## [1] 46.000000000 -0.001398271
## [1] 45.000000000 -0.001323028
## [1] 44.000000000 -0.001686916
## [1] 43.000000000 -0.0009328622
## [1] 42.000000000 -0.0009074548
## [1] 41.000000000 -0.0004498479
## [1] 40.000000000 -0.001203016
## [1] 39.000000000 -0.0004916008
## [1] 3.800000e+01 3.957131e-04
## [1] 37.000000000 -0.0004986884

```

```

## [1] 36.000000000 -0.001187701
## [1] 35.000000000 -0.0007220857
## [1] 3.40000e+01 4.58589e-05
## [1] 33.000000000 -0.0004193644
## [1] 32.000000000 -0.0001599588
## [1] 31.000000000 -0.0008791444
## [1] 3.00000e+01 3.625869e-04
## [1] 29.000000000 -0.001170846
## [1] 28.000000000 -0.001305727
## [1] 27.000000000 -0.001620862
## [1] 26.000000000 -0.001423234
## [1] 25.000000000 -0.0008829287
## [1] 2.40000e+01 1.489869e-04
## [1] 23.000000000 -0.0004989876
## [1] 2.20000e+01 -9.057882e-05
## [1] 2.10000e+01 6.718515e-04
## [1] 20.000000000 0.002724452
## [1] 19.000000000 0.001446484
## [1] 18.000000000 0.002233104
## [1] 17.000000000 0.003160998
## [1] 16.000000000 0.004158602
## [1] 15.0000000 0.00358678
## [1] 14.000000 0.0030625
## [1] 13.000000000 0.002573581
## [1] 12.000000000 0.004865908
## [1] 11.000000000 0.007943831
## [1] 10.000000000 0.007285574
## [1] 9.000000000 0.008718683
## [1] 8.000000000 0.007921909
## [1] 7.000000000 0.01051052

selection_results

## # A tibble: 70 x 3
##   `Number of Features` `AUC Loss` `Least Important Feature`
##   <int>      <dbl> <chr>
## 1 93 0 teste_esforco
## 2 75 0.00135 transplante_cardiaco
## 3 74 0.00125 transfusao
## 4 73 -0.000656 cardiac_arrest
## 5 72 0.000655 valvopathy
## 6 71 0.00140 insulina
## 7 70 -0.00552 flebografia
## 8 69 0.0000403 prior_mi
## 9 68 0.00142 teste_esforco
## 10 67 -0.000223 angio_tc
## # ... with 60 more rows

Minutes to run: 5.998

selected_features <- c(current_features, last_feature_dropped)

feature_selected_model <- model_fit_wf(selected_features,
                                         outcome_column, hyperparameters)

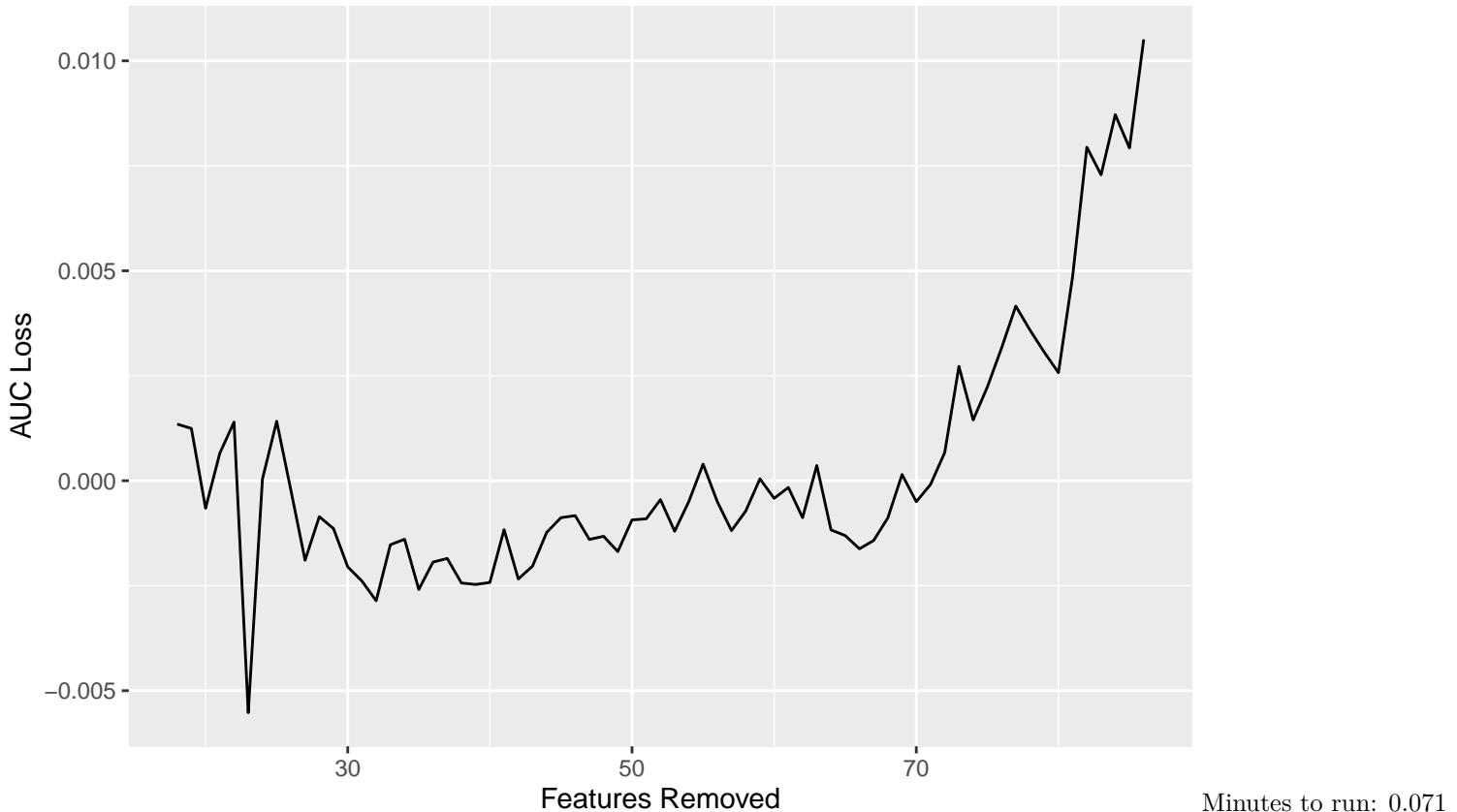
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.701"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.670"

```

```
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()
```



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)

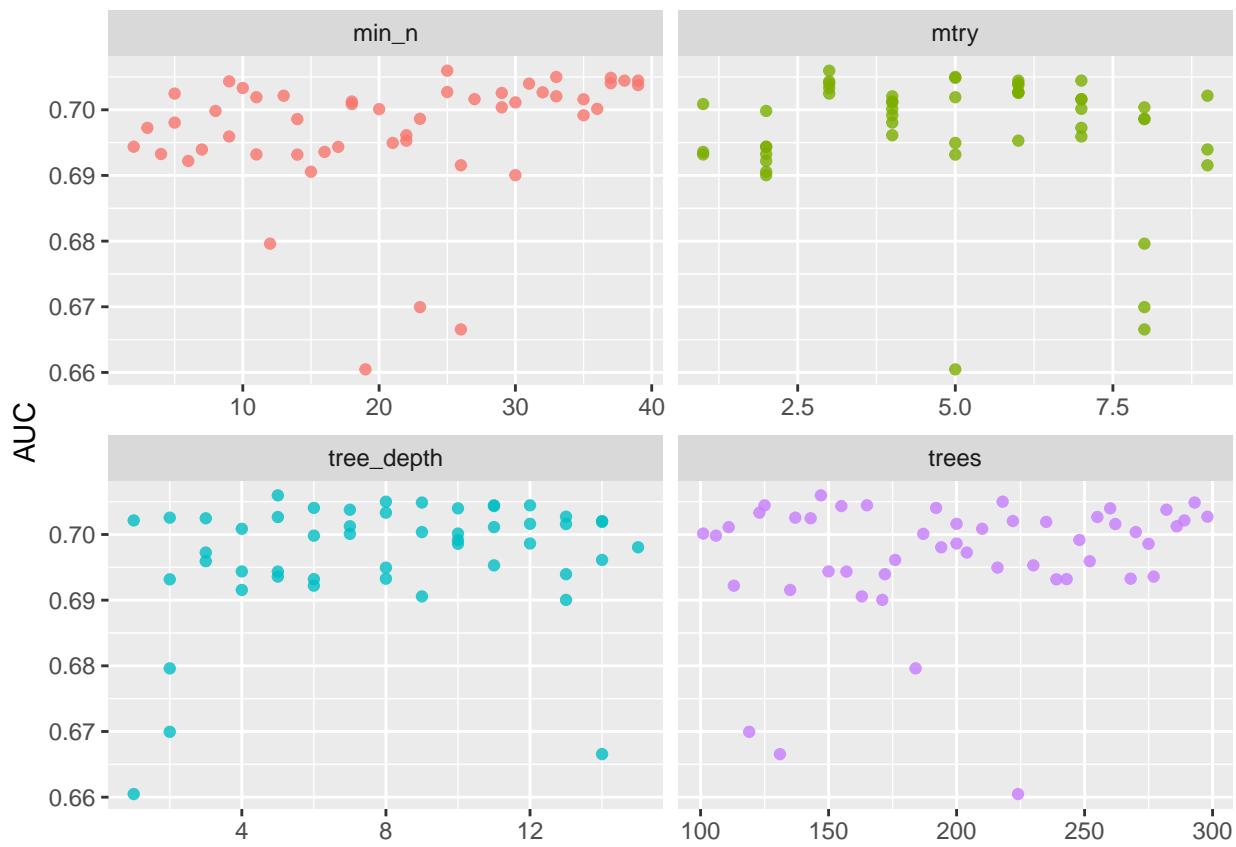
lightgbm_tune %>%
show_best("roc_auc")

## # 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     3    147     25        5  1.47e- 6  0.00000163 roc_auc binary  0.706    4  0.00589 Preprocessor1
## 2     5    218     33        8  6.97e- 6  0.891    roc_auc binary  0.705    4  0.00531 Preprocessor1
## 3     5    293     37        9  5.65e- 9  0.0000541 roc_auc binary  0.705    4  0.00535 Preprocessor1
## 4     7    165     39       11  1.47e- 2  7.03     roc_auc binary  0.704    4  0.00634 Preprocessor1
## 5     6    125     38       12  6.56e-10 0.0000220 roc_auc binary  0.704    4  0.00545 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")

```



```

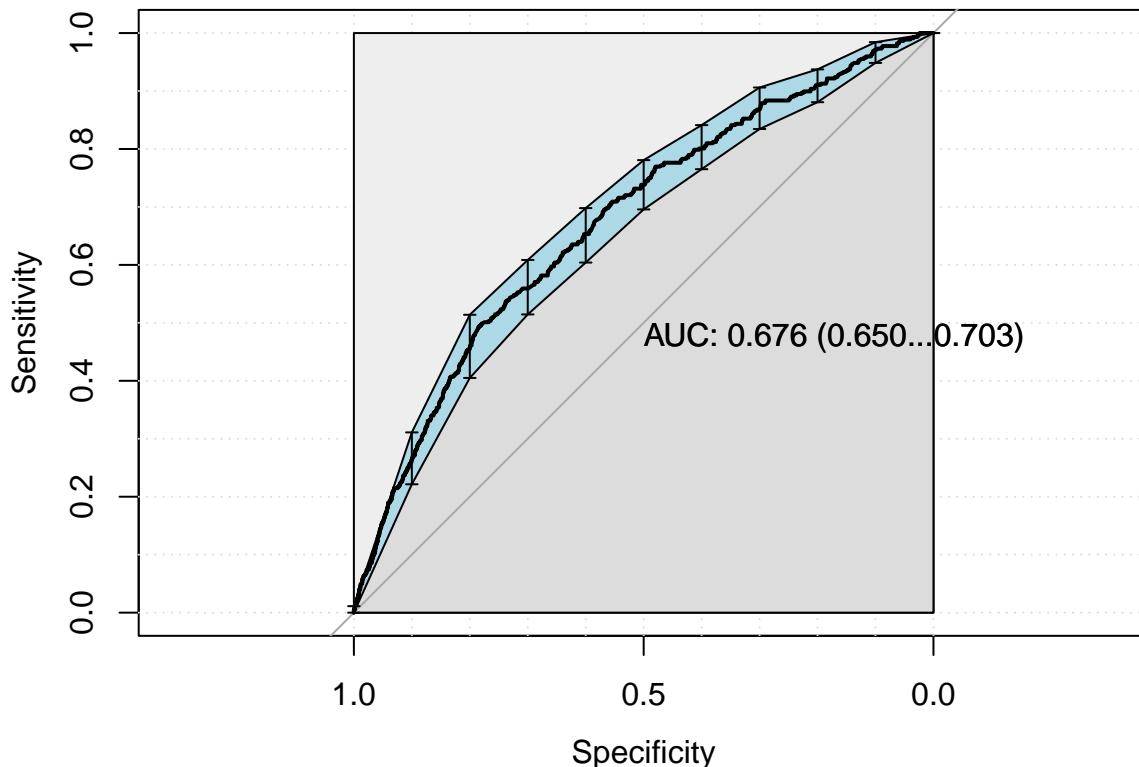
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4283  447
##                      1     0     0
## 
##                         Accuracy : 0.9055
##                         95% CI : (0.8968, 0.9137)
##    No Information Rate : 0.9055
##    P-Value [Acc > NIR] : 0.5126
## 
##                         Kappa : 0
## 
##    Mcnemar's Test P-Value : <2e-16
## 
##                         Sensitivity : 1.0000
##                         Specificity : 0.0000
##    Pos Pred Value : 0.9055
##    Neg Pred Value :      NaN
##    Prevalence : 0.9055
##    Detection Rate : 0.9055
##    Detection Prevalence : 1.0000
##    Balanced Accuracy : 0.5000
## 
##    '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: 1.379

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

```

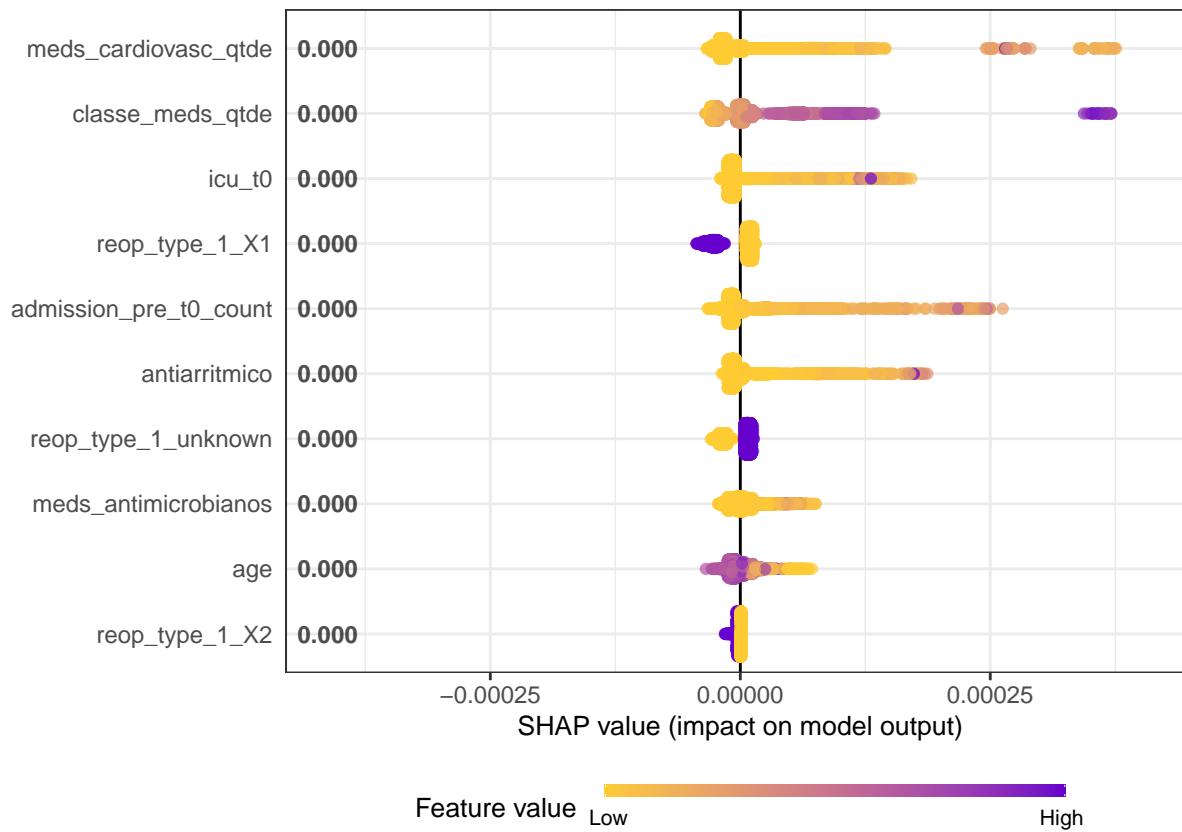
trained_rec <- prep(lightgbm_recipe, training = df_train)

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)

```

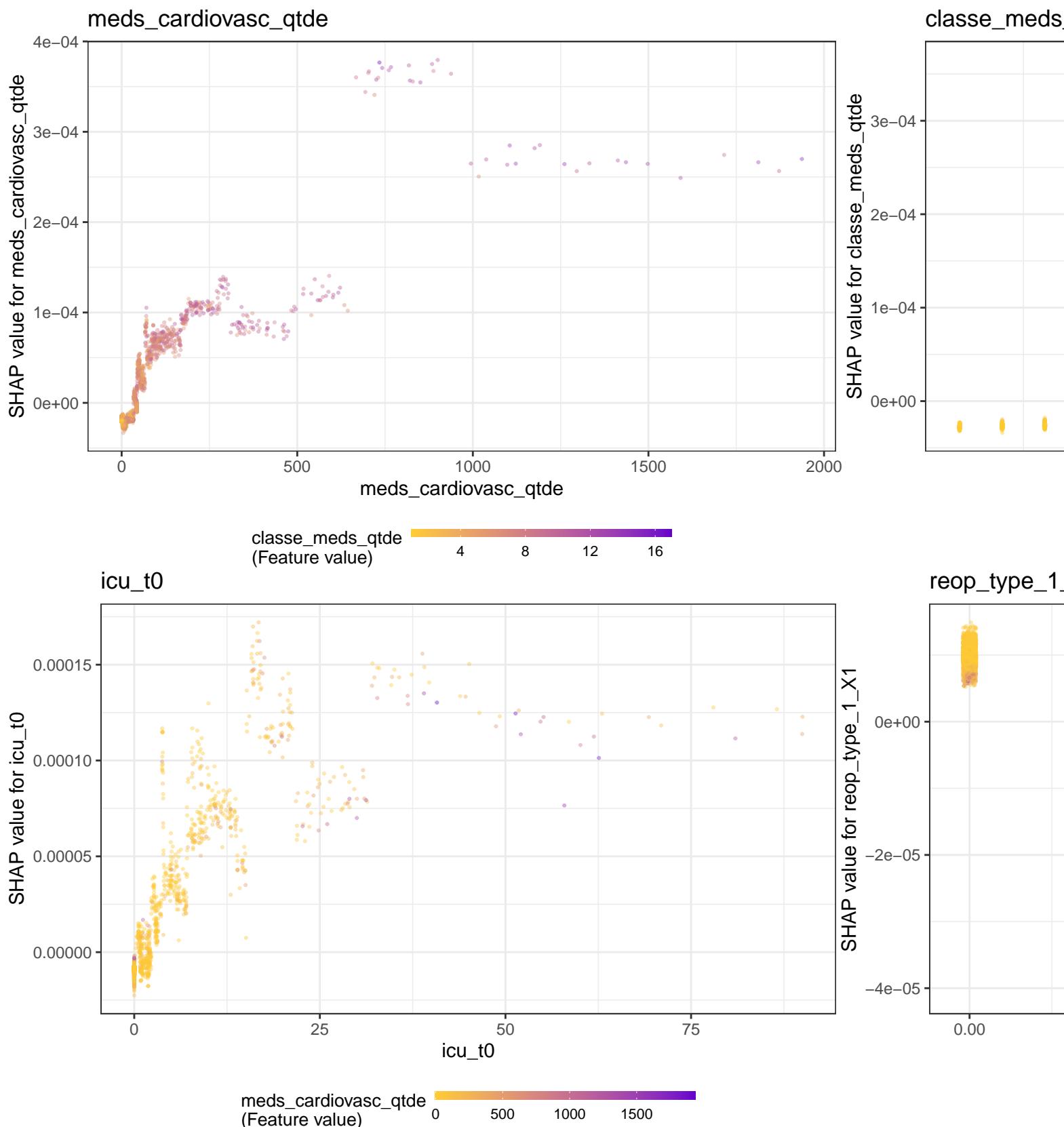


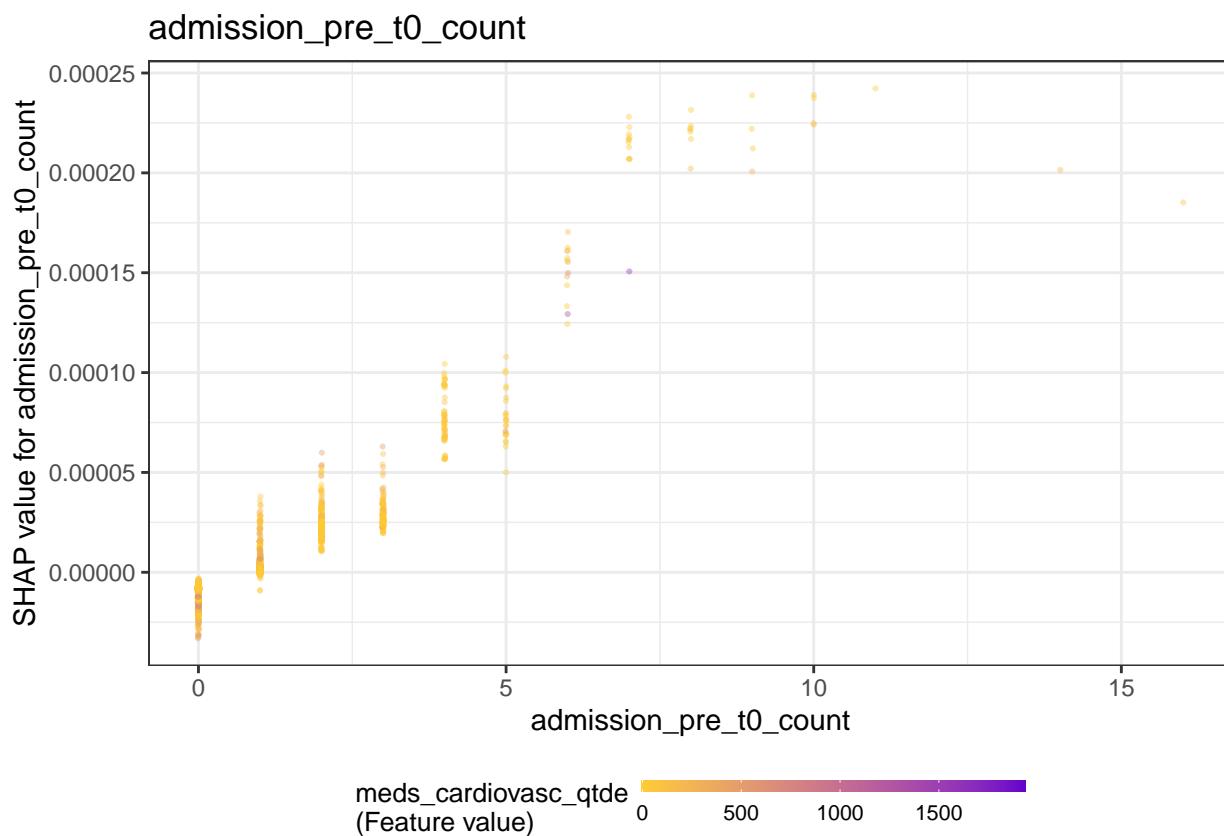
```

# 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)
}

```



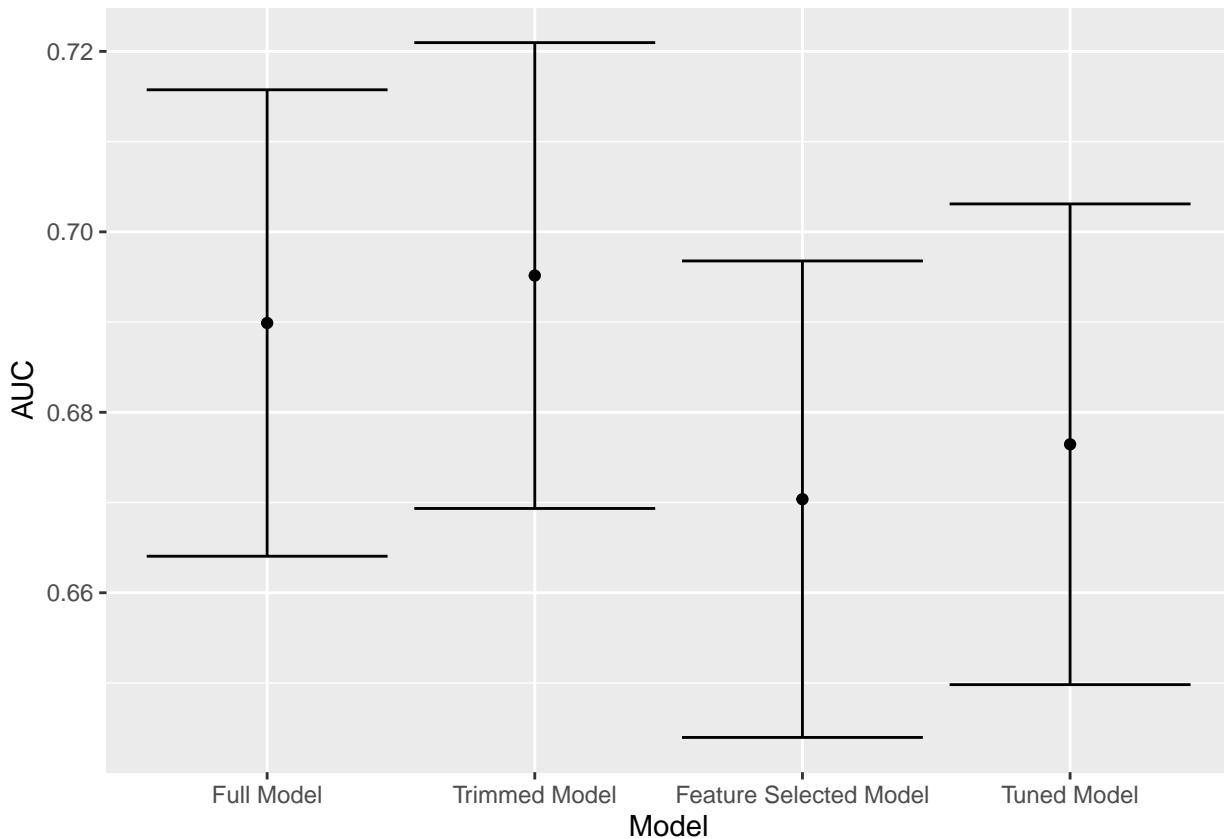


Minutes to run: 0

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$auc_upper,
  'Tuned Model', as.numeric(lightgbm_auc$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: 0.005