

Final Model - death_2year

Eduardo Yuki Yada

Imports

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

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

select <- dplyr::select
```

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

dir.create(file.path("./auxiliar/final_model/hyperparameters/"),
          showWarnings = FALSE,
          recursive = TRUE)

dir.create(file.path("./auxiliar/final_model/performance/"),
          showWarnings = FALSE,
          recursive = TRUE)

dir.create(file.path("./auxiliar/final_model/selected_features/"),
          showWarnings = FALSE,
          recursive = TRUE)
```

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)
}

```

Starting features:

```
gluedown::md_order(features, seq = TRUE, pad = TRUE)
```

1. sex
2. age
3. education_level
4. underlying_heart_disease
5. heart_disease
6. nyha_basal
7. hypertension
8. prior_mi
9. heart_failure
10. af
11. cardiac_arrest
12. valvopathy
13. diabetes
14. renal_failure
15. hemodialysis
16. stroke
17. copd
18. comorbidities_count
19. procedure_type_1
20. reop_type_1
21. procedure_type_new
22. cied_final_1
23. cied_final_group_1
24. admission_pre_t0_count
25. admission_pre_t0_180d
26. year_adm_t0
27. icu_t0
28. dialysis_t0
29. admission_t0_emergency
30. aco
31. antiarritmico
32. ieca_bra
33. dva
34. digoxina
35. estatina
36. diuretico
37. vasodilatador

38. insuf_cardiaca
39. espironolactona
40. antiplaquetario_ev
41. insulina
42. psicofarmacos
43. antifungico
44. antiviral
45. classe_meds_qtde
46. meds_cardiovasc_qtde
47. meds_antimicrobianos
48. vni
49. outros_proced_cirurgicos
50. icp
51. angioplastia
52. cateterismo
53. cateter_venoso_central
54. proced_invasivos_qtde
55. transfusao
56. interconsulta
57. equipe_multiprof
58. ecg
59. holter
60. teste_esforco
61. tilt_teste
62. metodos_graficos_qtde
63. laboratorio
64. cultura
65. analises_clinicas_qtde
66. citologia
67. histopatologia_qtde
68. angio_tc
69. cintilografia
70. ecocardiograma
71. endoscopia
72. flebografia
73. pet_ct
74. ultrassom
75. tomografia
76. radiografia
77. ressonancia
78. exames_imagem_qtde
79. bic

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) %>% select(all_of(c(features, outcome_column)))
df_test <- testing(df_split) %>% select(all_of(c(features, outcome_column)))
```

Global parameters

```
k <- 5 # 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
```

Functions

```
nicerFormatting <- function(df, caption="", digits = 2, font_size = NULL){
  df %>%
    kbl(booktabs = T, longtable = T, caption = caption,
        digits = digits, format = "latex") %>%
    kable_styling(font_size = font_size,
                  latex_options = c("striped", "HOLD_position", "repeat_header"))
}

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)
}

```

Feature Selection

```

model_fit_wf <- function(df_train, 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]
    )
  )
}

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

full_model <- model_fit_wf(df_train, features, outcome_column, hyperparameters)

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

```

```

## [1] "Full Model CV Train AUC: 0.771"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.792"

Features with zero importance on the initial model:
unimportant_features <- setdiff(features, full_model$importance$Feature)

unimportant_features %>%
  gluedown::md_order()

1. hemodialysis
2. vni
3. angioplastia
4. transfusao
5. teste_esforco
6. histopatologia_qtde

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

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

## [1] "Trimmed Model CV Train AUC: 0.770"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.790"

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

if (full_model$cv_auc - trimmed_model$cv_auc < max_auc_loss) {
  current_features <- trimmed_features
  current_model <- trimmed_model
  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(trimmed_features),
            `AUC Loss` = current_auc_loss,
            `Least Important Feature` = current_least_important)
} else {
  current_features <- features
  current_model <- full_model
  current_least_important <- tail(current_model$importance$Feature, 1)
  current_auc_loss <- 0
}

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(df_train, 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))
}

selection_results %>% niceFormatting(digits = 4)

```

Table 1:

Number of Features	AUC Loss	Least Important Feature
79	0.0000	angio_tc
73	0.0009	pet_ct
72	0.0007	copd
71	0.0011	antiviral
70	0.0011	flebografia
69	0.0005	angio_tc
68	0.0022	tilt_teste
67	0.0013	antifungico
66	0.0009	cateter_venoso_central
65	0.0003	dialysis_t0
64	0.0012	underlying_heart_disease
63	0.0018	ressonancia
62	0.0014	procedure_type_new
61	0.0027	endoscopia
60	0.0023	icp
59	0.0021	cateterismo
58	0.0018	diabetes
57	0.0026	holter
56	0.0028	reop_type_1
55	0.0017	antiplaquetario_ev
54	0.0016	cied_final_1
53	0.0035	aco
52	0.0030	insulina
51	0.0043	stroke
50	0.0037	citologia
49	0.0041	sex
48	0.0032	outros_proced_cirurgicos
47	0.0032	cardiac_arrest
46	0.0025	valvopathy
45	0.0033	heart_failure
44	0.0019	proced_invasivos_qtde
43	0.0006	bic
42	0.0008	tomografia
41	-0.0002	renal_failure
40	0.0005	ecocardiograma
39	0.0008	cultura
38	0.0016	nyha_basal
37	0.0057	admission_t0_emergency
36	0.0039	cintilografia
35	0.0047	interconsulta
34	0.0048	heart_disease
33	0.0048	hypertension
32	0.0048	prior_mi
31	0.0042	ultrassom

Table 1: (*continued*)

Number of Features	AUC Loss	Least Important Feature
30	0.0036	digoxina
29	0.0034	insuf_cardiaca
28	0.0047	procedure_type_1
27	0.0058	af
26	0.0066	radiografia
25	0.0072	dva
24	0.0101	admission_pre_t0_180d

```

if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_dropped)
} else {
  selected_features <- current_features
}

con <- file(sprintf('./auxiliar/final_model/selected_features/%s.yaml', outcome_column), "w")
write_yaml(selected_features, con)
close(con)

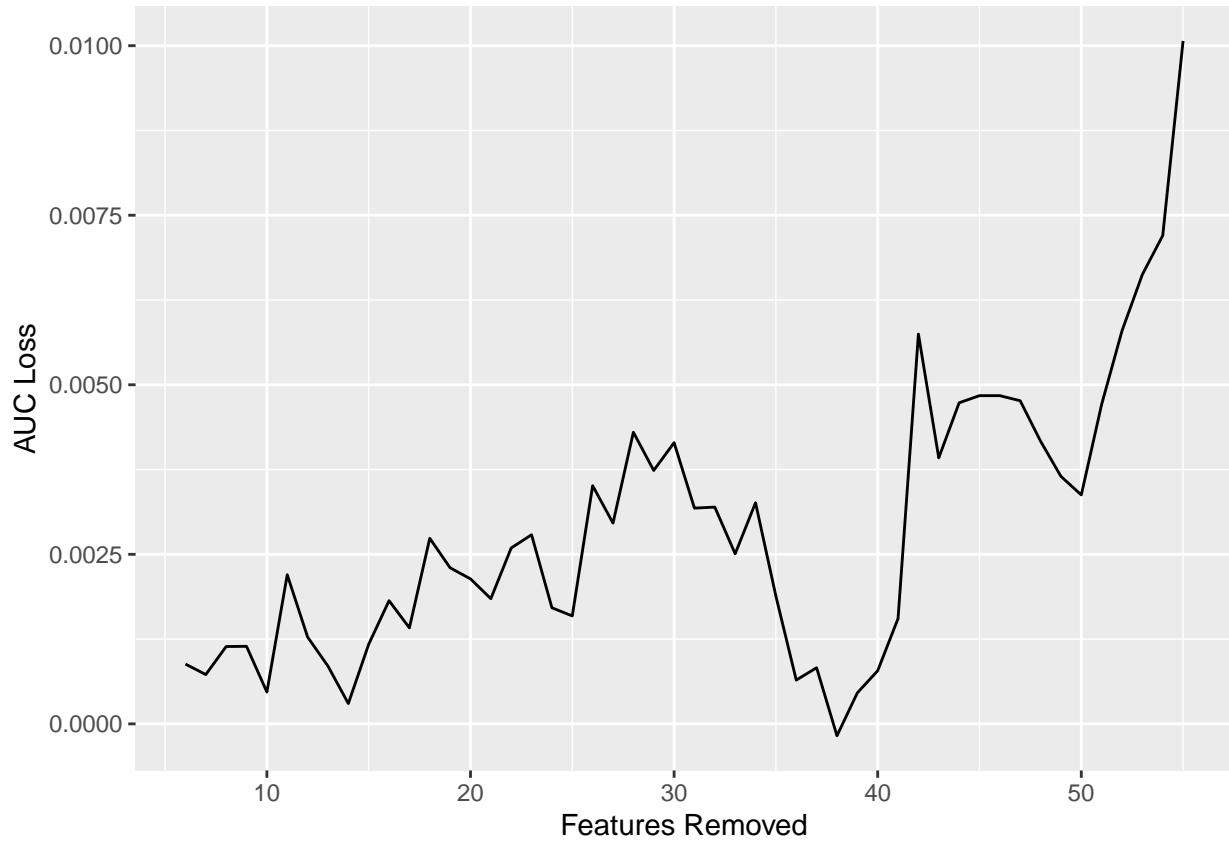
feature_selected_model <- model_fit_wf(df_train, selected_features,
                                         outcome_column, hyperparameters)

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

## [1] "Trimmed Model CV Train AUC: 0.761"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.777"
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

Selected features:

```
gluedown::md_order(selected_features, seq = TRUE, pad = TRUE)
```

1. age
2. admission_pre_t0_count
3. year_adm_t0
4. laboratorio
5. espironolactona
6. analises_clinicas_qtde
7. comorbidities_count
8. vasodilatador
9. icu_t0
10. ecg
11. meds_cardiovasc_qtde
12. meds_antimicrobianos
13. cied_final_group_1
14. equipe_multiprof
15. education_level
16. diuretico
17. psicofarmacos
18. ieca_bra
19. antiarritmico
20. metodos_graficos_qtde
21. classe_meds_qtde
22. estatina
23. exames_imagem_qtde
24. admission_pre_t0_180d

```
# doParallel::registerDoParallel(8)
```

Smote

```

library(themis)

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

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") %>%
  niceFormatting(digits = 5)

```

Table 2:

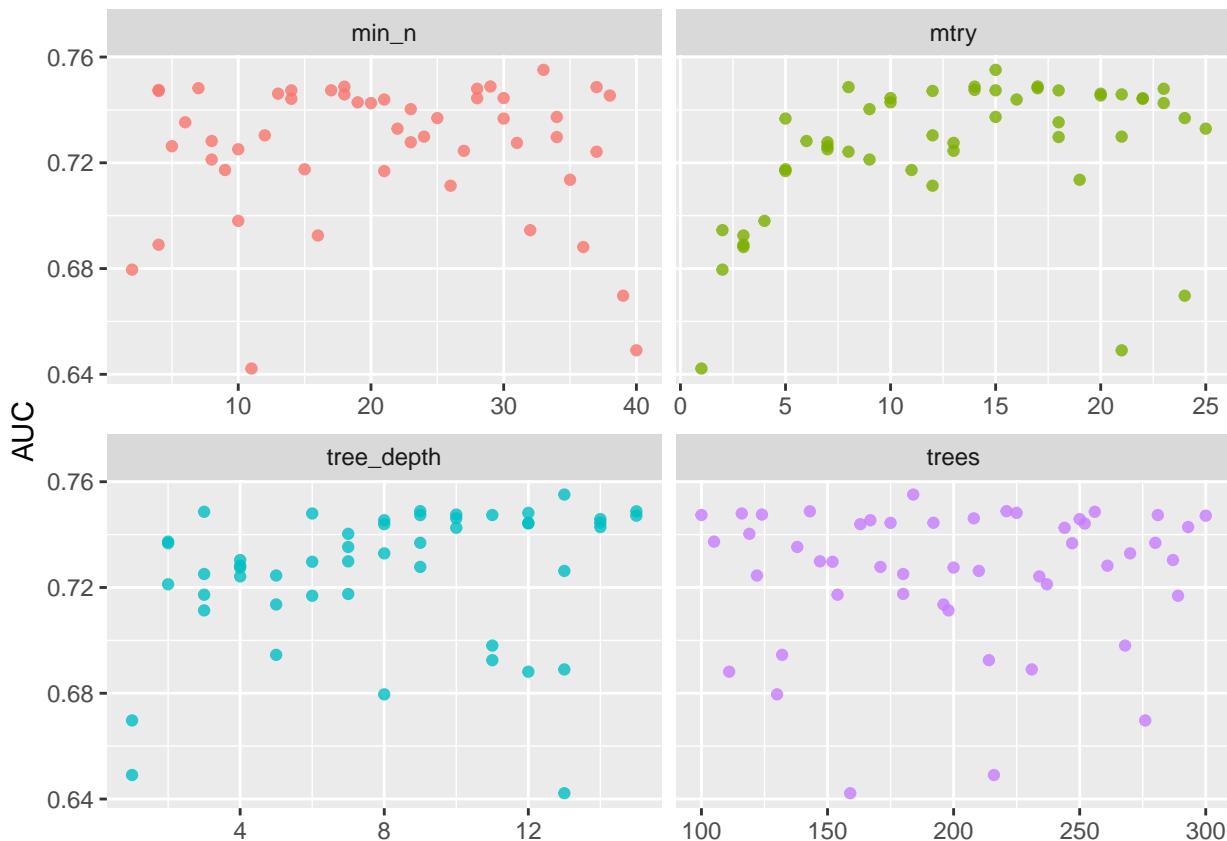
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
15	184	33	13	0.00660	0.00000	roc_auc	binary	0.75515	5	0.00400	Preprocessor1
17	221	29	9	0.00000	23.38609	roc_auc	binary	0.74887	5	0.00312	Preprocessor1
14	143	18	15	0.00000	0.00000	roc_auc	binary	0.74881	5	0.00414	Preprocessor1
8	256	37	3	0.07272	9.10365	roc_auc	binary	0.74863	5	0.00407	Preprocessor1
17	225	7	12	0.00000	0.00000	roc_auc	binary	0.74822	5	0.00189	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")

```



```

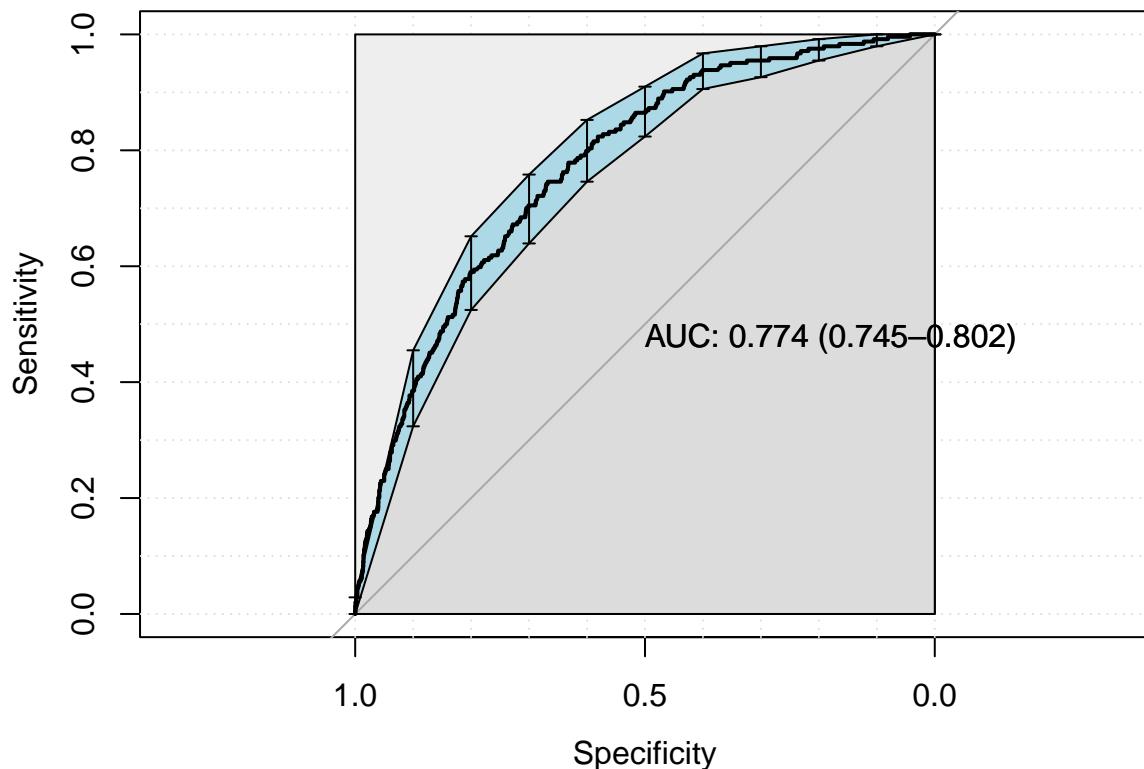
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_smote_auc <- validation(final_lightgbm_fit, df_test)

```



```

## Confusion Matrix and Statistics
##
## test_predictions_class      0      1
##                      0 4439  229
##                      1   47   15
##
##          Accuracy : 0.9416
## 95% CI : (0.9346, 0.9482)
##  No Information Rate : 0.9484
## P-Value [Acc > NIR] : 0.9823
##
##          Kappa : 0.0788
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.98952
##          Specificity  : 0.06148
##  Pos Pred Value  : 0.95094
##  Neg Pred Value  : 0.24194
##          Prevalence  : 0.94841
##  Detection Rate  : 0.93848
## Detection Prevalence : 0.98689
##  Balanced Accuracy : 0.52550
##
##  '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

saveRDS(

```

```

    lightgbm_parameters,
    file = sprintf(
      "./auxiliar/final_model/hyperparameters/lightgbm_smote_%s.rds",
      outcome_column
    )
)

```

Upsample

```

lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% 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()) %>%
  step_upsample (!!sym(outcome_column))

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") %>%
  niceFormatting(digits = 5)

```

Table 3:

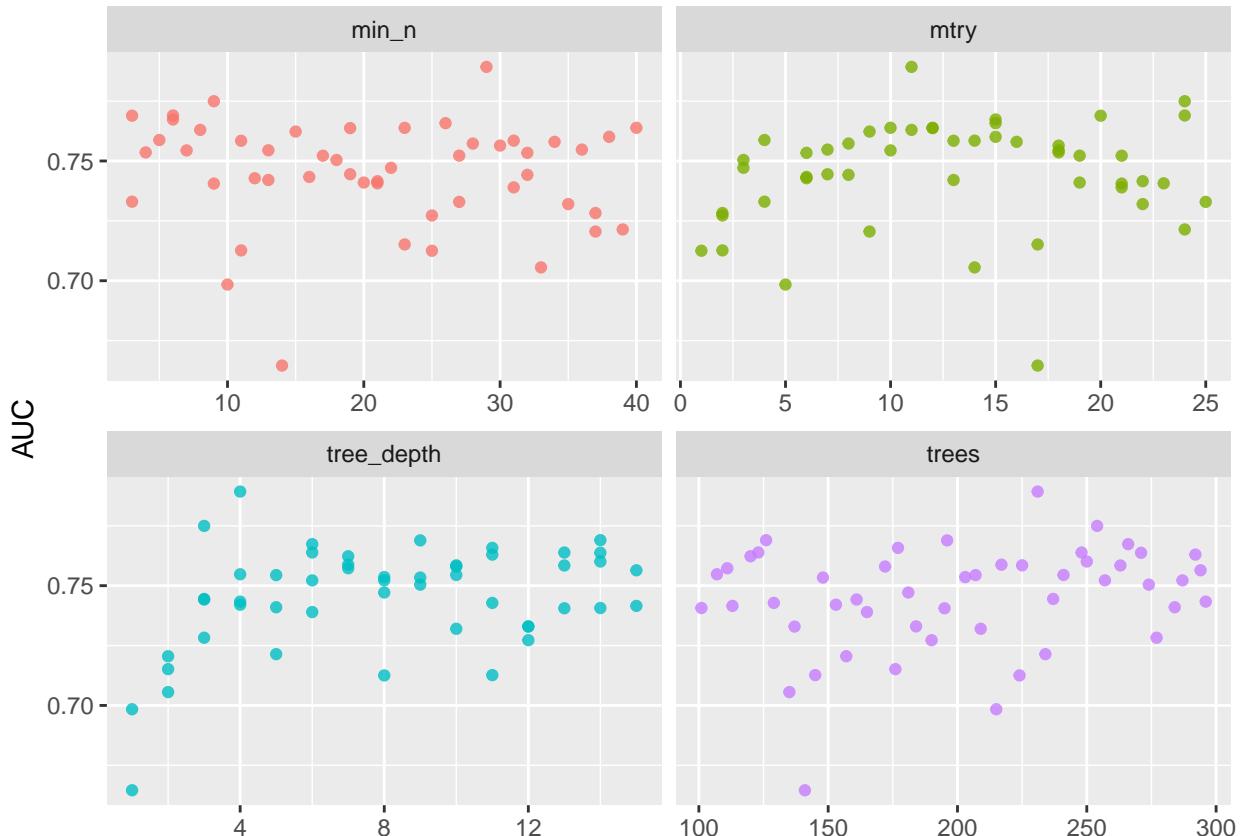
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
11	231	29	4	0.07574	29.92048	roc_auc	binary	0.78928	5	0.00693	Preprocessor1

Table 3: (continued)

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
24	254	9	3	0.01081	0.51752	roc_auc	binary	0.77495	5	0.00647	Preprocessor1
24	126	6	14	0.02088	0.00000	roc_auc	binary	0.76900	5	0.00751	Preprocessor1
20	196	3	9	0.01853	0.00149	roc_auc	binary	0.76891	5	0.00861	Preprocessor1
15	266	6	6	0.00605	0.00000	roc_auc	binary	0.76734	5	0.00975	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)

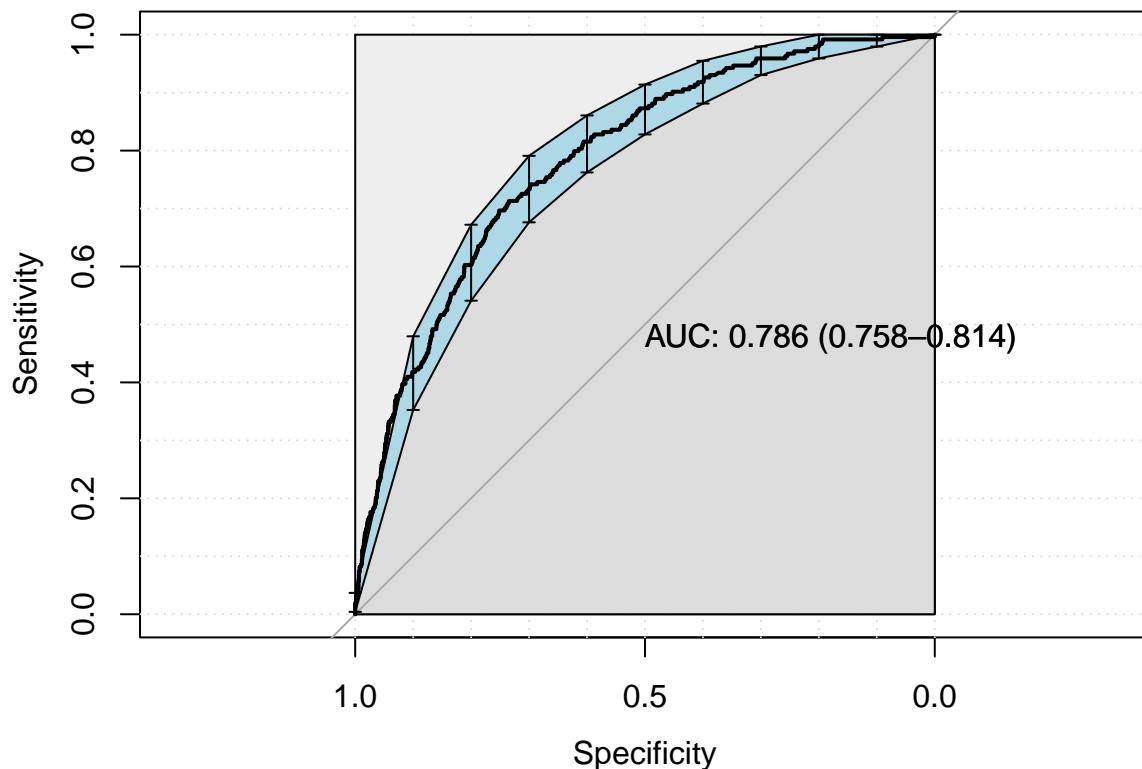
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)
```

```

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_upsample_auc <- validation(final_lightgbm_fit, df_test)

```



```

## Confusion Matrix and Statistics
##
## test_predictions_class      0      1
##                      0 3452   81
##                      1 1034  163
##
##          Accuracy : 0.7643
## 95% CI : (0.7519, 0.7763)
## No Information Rate : 0.9484
## P-Value [Acc > NIR] : 1
##
##          Kappa : 0.1537
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.7695
##          Specificity  : 0.6680
## Pos Pred Value : 0.9771
## Neg Pred Value : 0.1362
##          Prevalence : 0.9484
## Detection Rate : 0.7298
## Detection Prevalence : 0.7469
## Balanced Accuracy : 0.7188
##
## '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

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

```

Standard

```

lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% 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") %>%
  niceFormatting(digits = 5)

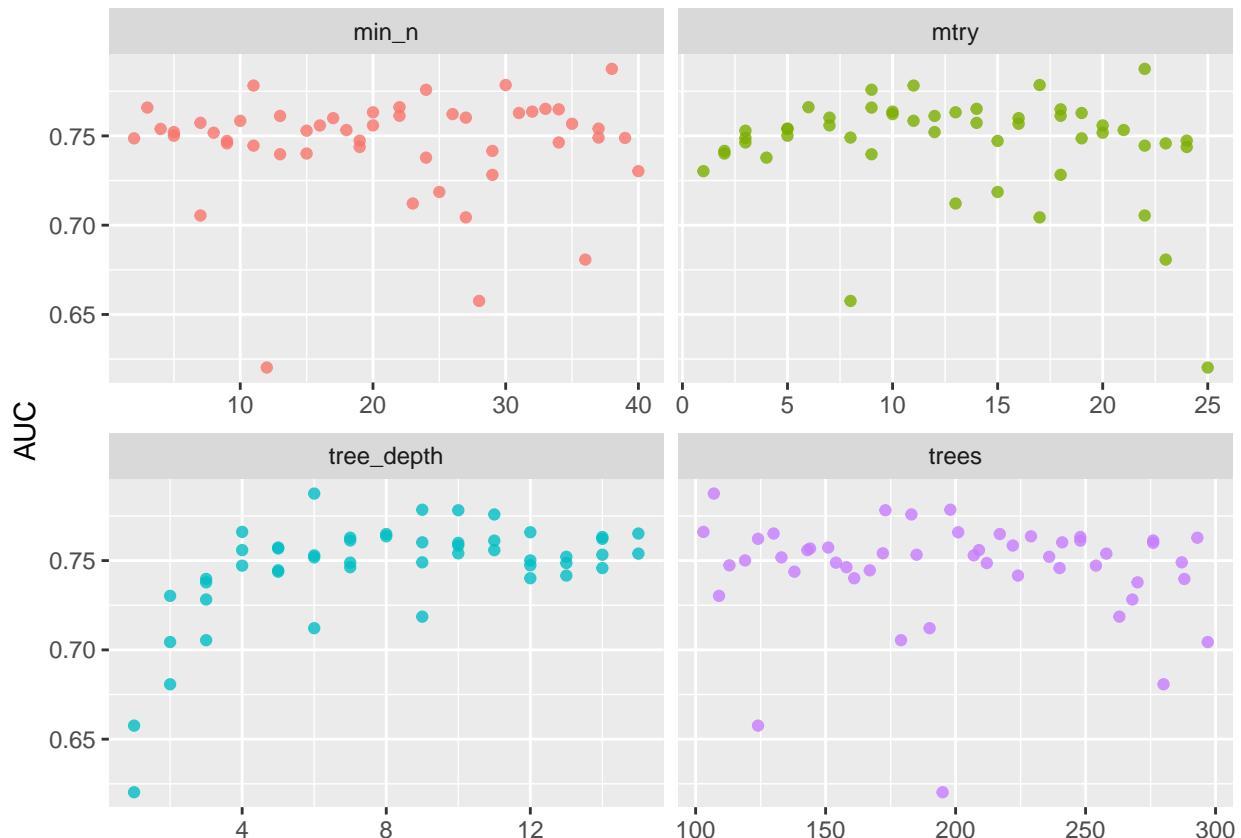
```

Table 4:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
22	107	38	6	0.02859	0.00000	roc_auc	binary	0.78754	5	0.00496	Preprocessor1
17	198	30	9	0.00775	0.00092	roc_auc	binary	0.77850	5	0.00773	Preprocessor1
11	173	11	10	0.03400	0.00034	roc_auc	binary	0.77820	5	0.00581	Preprocessor1
9	183	24	11	0.00978	0.31523	roc_auc	binary	0.77584	5	0.00639	Preprocessor1
6	103	22	4	0.01863	0.00000	roc_auc	binary	0.76608	5	0.01225	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)

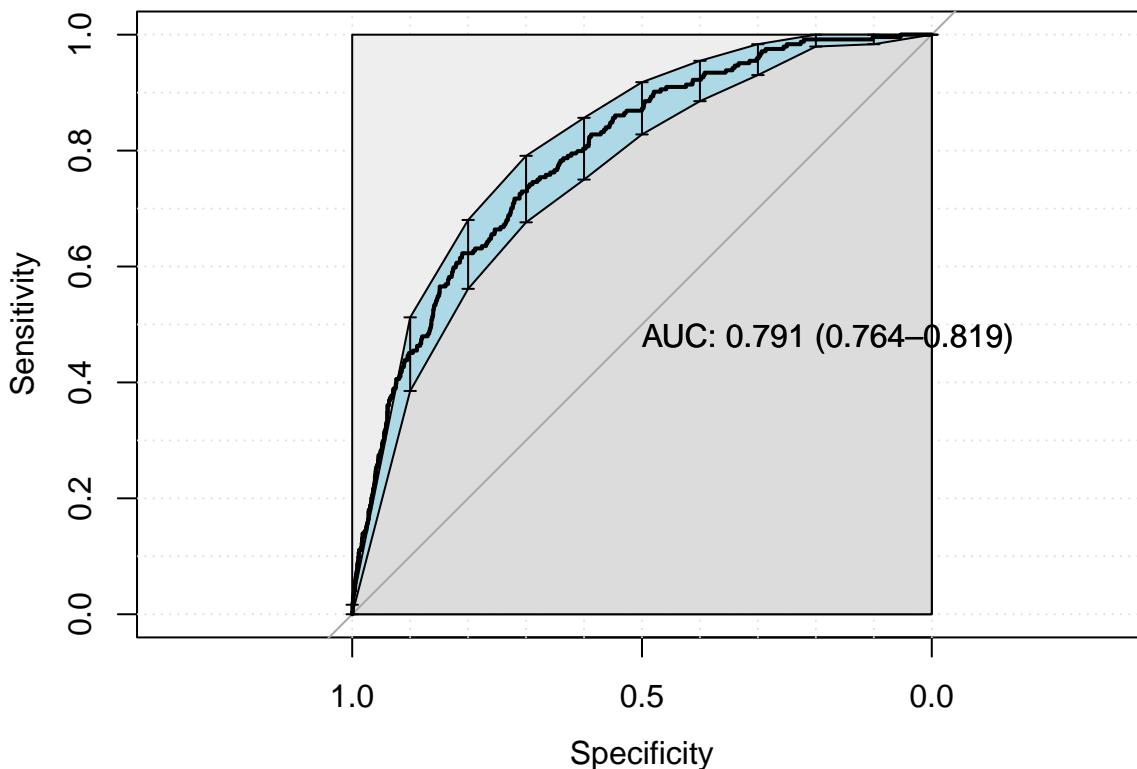
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)
```

```

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 4482  241
##                      1     4     3
##
##          Accuracy : 0.9482
## 95% CI : (0.9415, 0.9543)
## No Information Rate : 0.9484
## P-Value [Acc > NIR] : 0.5431
##
##          Kappa : 0.0211
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.9991
##          Specificity  : 0.0123
## Pos Pred Value  : 0.9490
## Neg Pred Value  : 0.4286
##          Prevalence  : 0.9484
## Detection Rate  : 0.9476
## Detection Prevalence : 0.9985
## Balanced Accuracy : 0.5057
##
## '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

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

```

SHAP values

```

lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

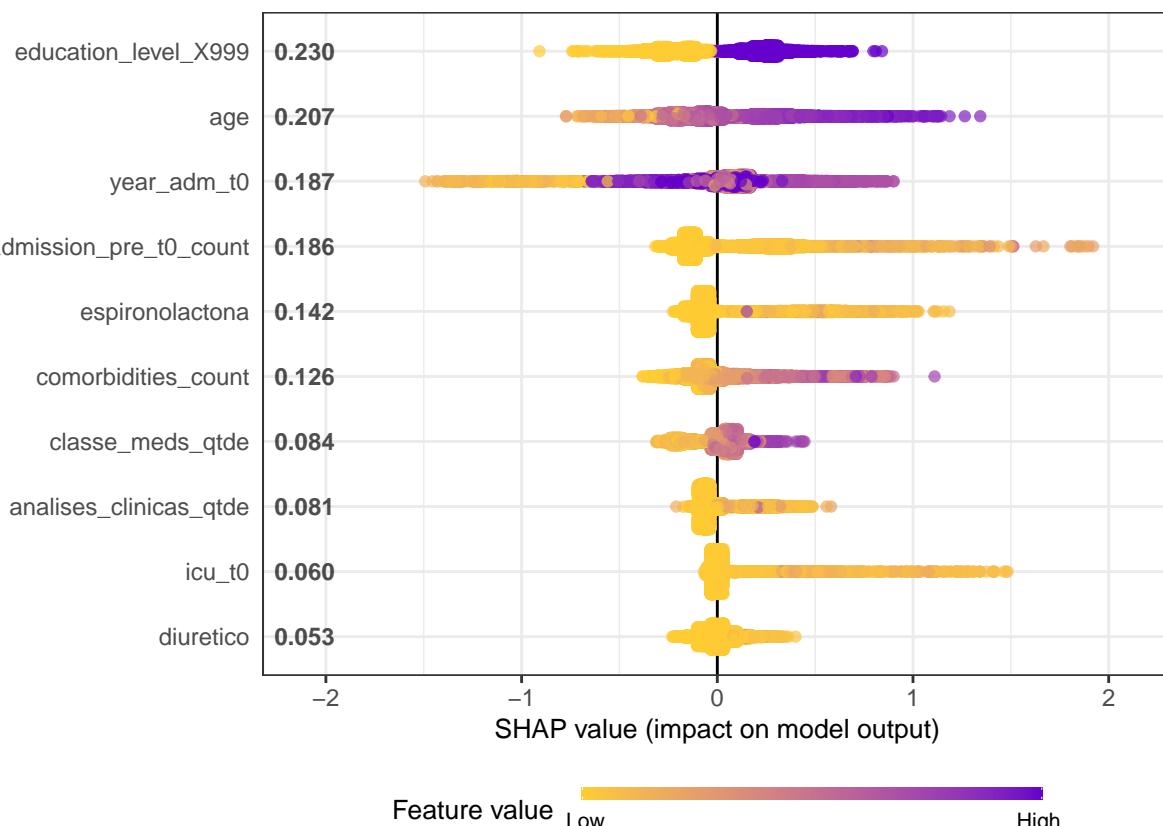
trained_rec <- prep(lightgbm_recipe, training = df_train)

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 = TRUE,
jitter_width = 0.01,
alpha = 0.3
) +
  labs(title = x)
print(p)
}

## `geom_smooth()` using formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -0.005

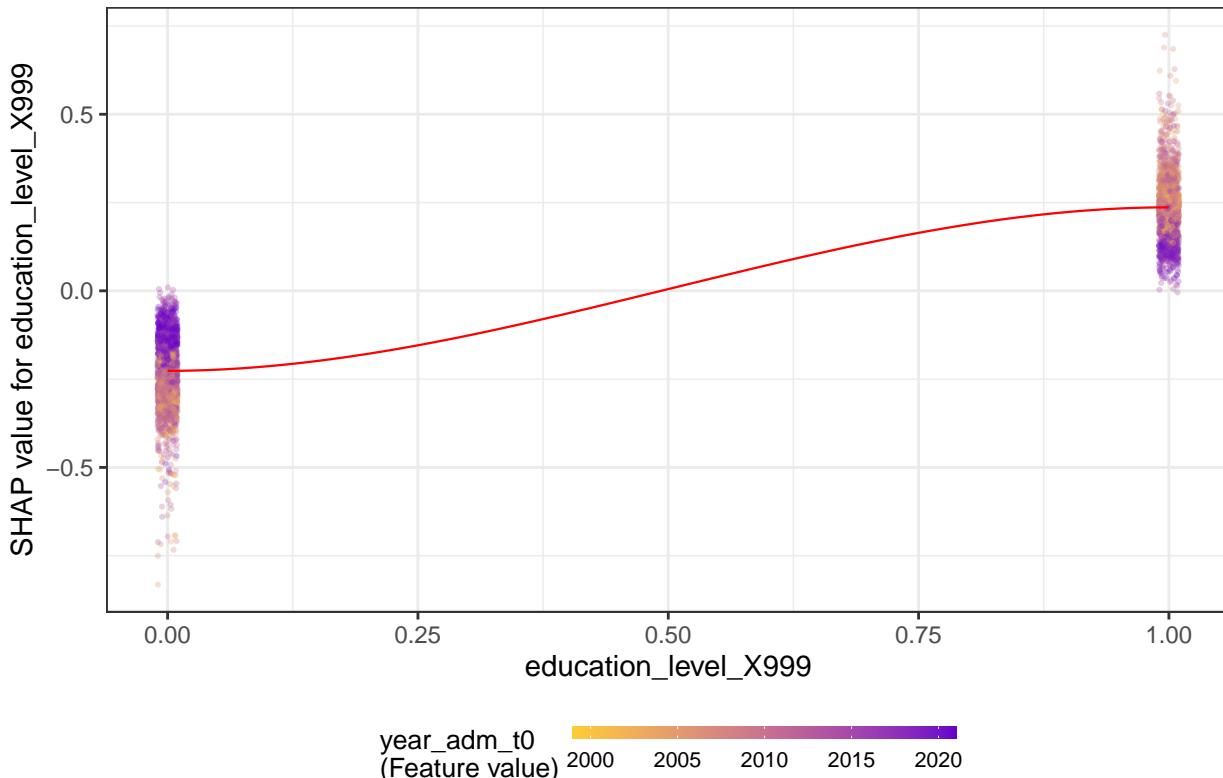
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 1.5385e-29

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are
## other near singularities as well. 1.01

```

education_level_X999

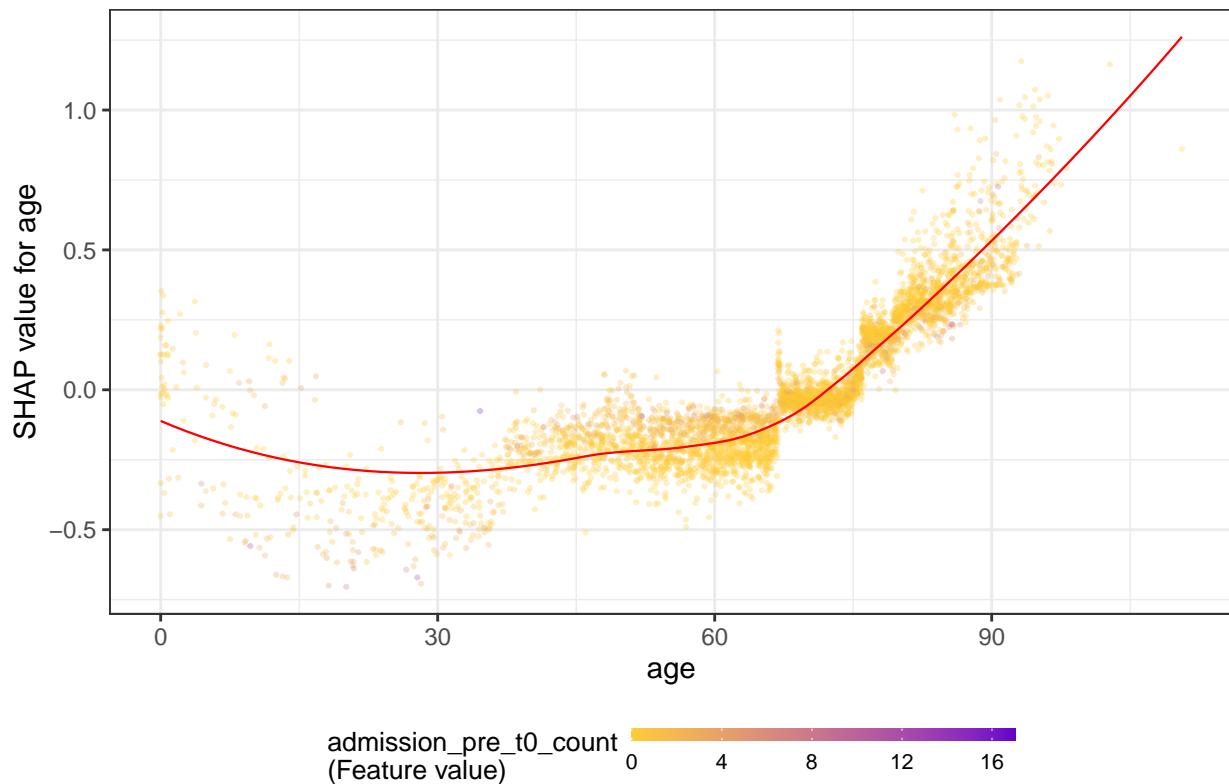


```

## `geom_smooth()` using formula 'y ~ x'

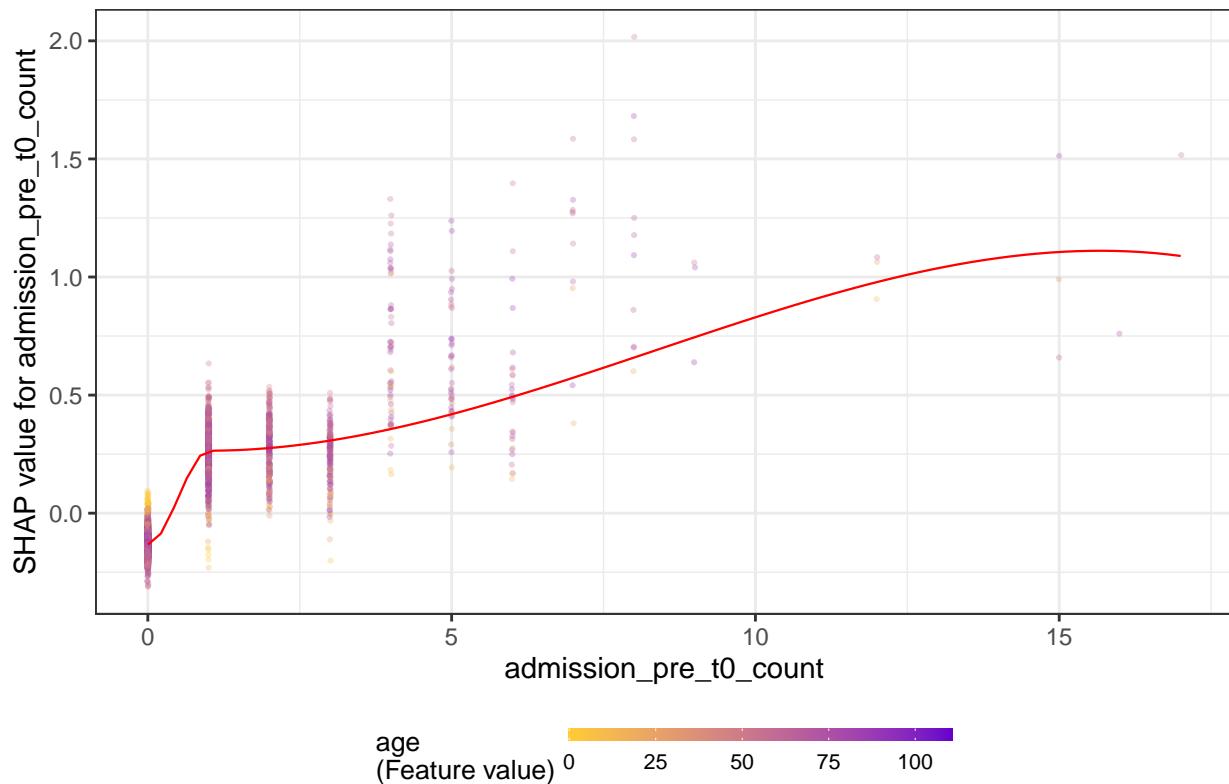
```

age



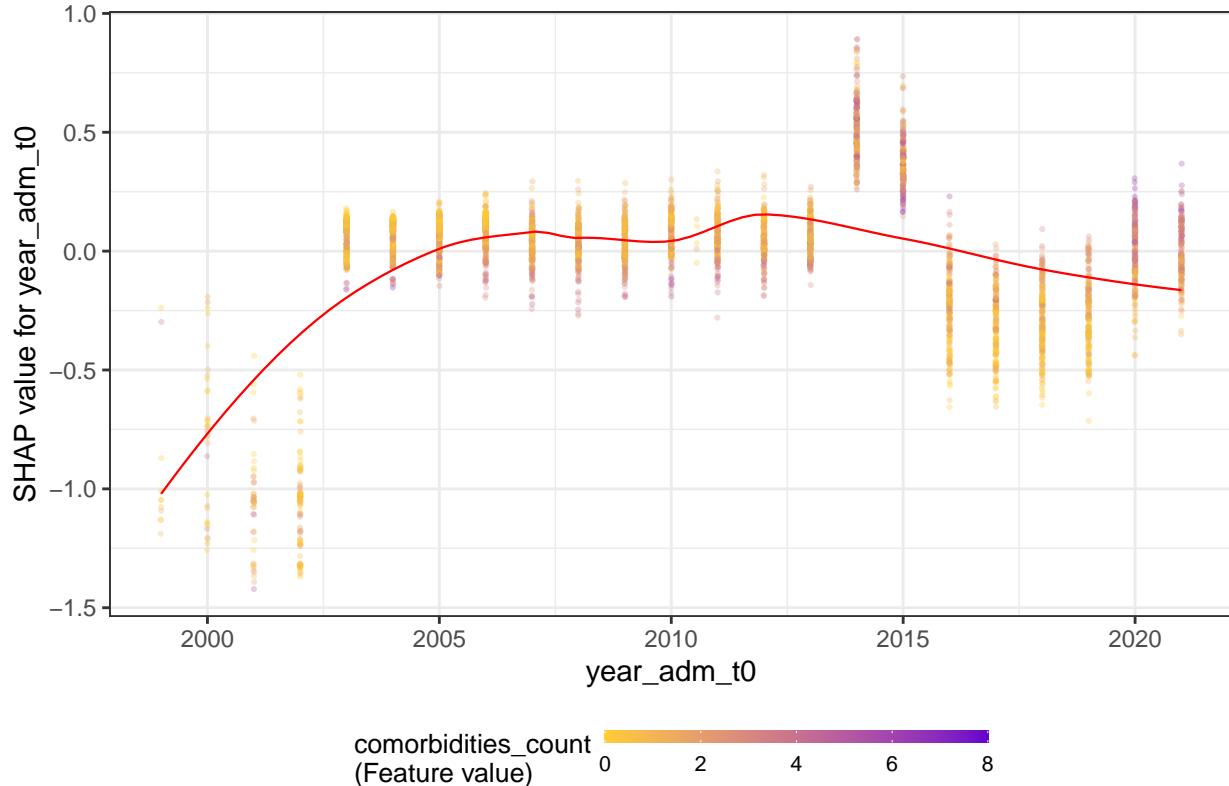
```
## `geom_smooth()` using formula 'y ~ x'  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse  
## used at -0.085  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood  
## radius 1.085  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal  
## condition number 1.7903e-27  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are  
## other near singularities as well. 1
```

admission_pre_t0_count



```
## `geom_smooth()` using formula 'y ~ x'
```

year_adm_t0



```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -1.02
```

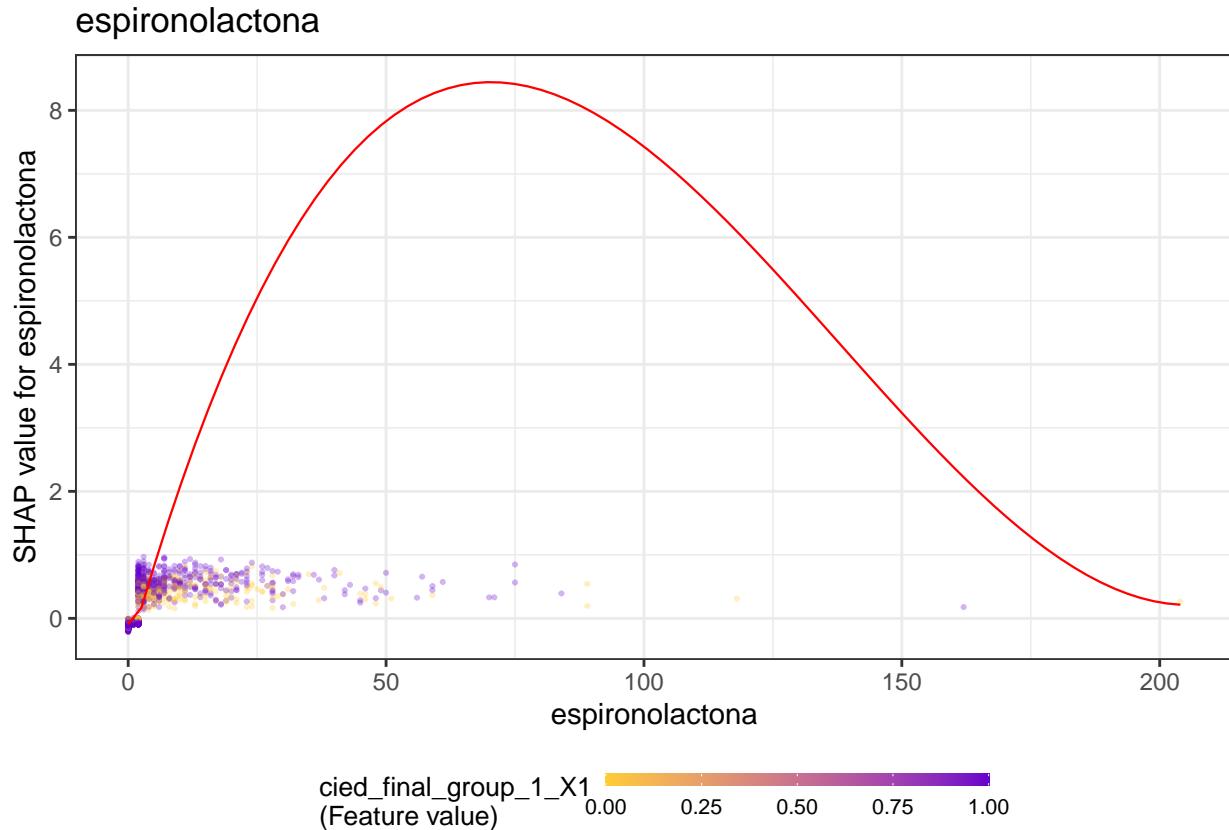
```

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 2.9806

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 1.3267e-14

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are
## other near singularities as well. 3.8441

```



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$auc_upper,
  'Tuned Standard Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3],
  'Tuned Smote Model', as.numeric(lightgbm_smote_auc$auc), lightgbm_smote_auc$ci[1], lightgbm_smote_auc$ci[3],
  'Tuned Upsample Model', as.numeric(lightgbm_upsample_auc$auc), lightgbm_upsample_auc$ci[1], lightgbm_upsample_auc$ci[3]
) %>%
  mutate(Target = outcome_column,
        Model = factor(Model,
                        levels = c('Full Model', 'Trimmed Model',
                                  'Feature Selected Model', 'Tuned Standard Model',
                                  'Tuned Smote Model', 'Tuned Upsample Model')))

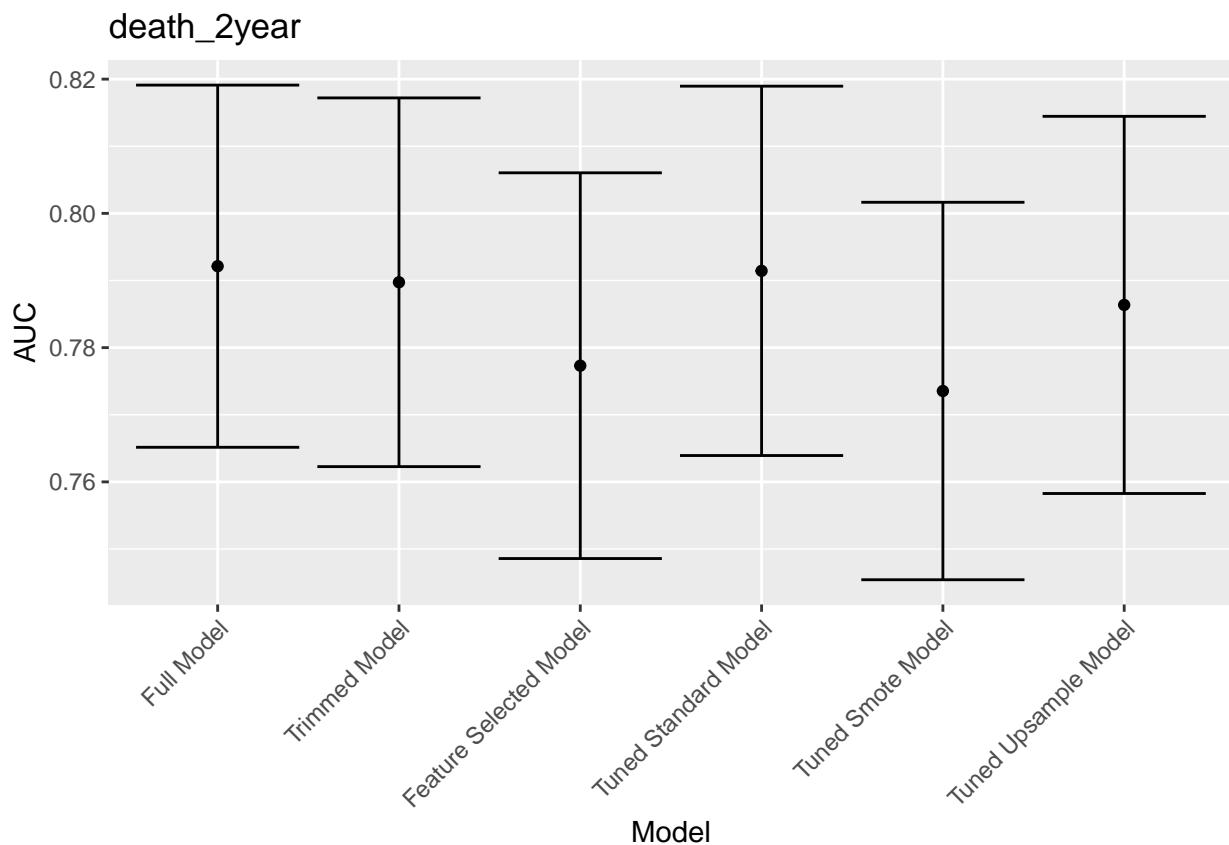
df_auc %>%
  ggplot(aes(
    x = Model,
    y = AUC,
    ymin = `Lower Limit`,
    ymax = `Upper Limit`)

```

```

)) +
geom_point() +
geom_errorbar() +
labs(title = outcome_column) +
theme(axis.text.x = element_text(angle = 45, hjust=1))

```



```
saveRDS(df_auc, sprintf("./auxiliar/final_model/performance/%s.RData", outcome_column))
```