Final Model - death_180days

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. cancer
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. year_adm_t0
28. icu_t0
29. dialysis_t0
30. admission_t0_emergency
31. aco
32. antiarritmico
33. ieca_bra
34. dva
35. digoxina
36. estatina
37. diuretico

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

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

max_auc_loss <- 0.01
```

Functions

```r
niceFormatting <- 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"))
}
```

```r
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: 0.776"
```

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

```
## [1] "Full Model Test AUC: 0.847"
```

Features with zero importance on the initial model:

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

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

1. dialysis_t0

```
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: 0.783"
```

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

```
## [1] "Trimmed Model Test AUC: 0.850"
```

```
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_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(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
77	0.0000	arteriografia
76	-0.0072	teste_esforco
75	-0.0087	aortografia
74	-0.0057	cateter_venoso_central
73	-0.0027	vni
72	-0.0003	angiografia
71	0.0002	transfusao
70	0.0022	arteriografia
69	0.0000	heart_disease
68	-0.0032	histopatologia_qtde
67	-0.0011	antifungico
66	-0.0029	outros_proced_cirurgicos
65	-0.0024	icp
64	-0.0026	underlying_heart_disease
63	-0.0025	holter
62	0.0019	cied_final_1
61	-0.0004	valvopathy
60	-0.0006	cardiac_arrest
59	-0.0034	hemodialysis
58	-0.0016	endoscopia
57	-0.0055	procedure_type_1
56	-0.0050	bic
55	0.0036	cancer
54	-0.0008	digoxina
53	-0.0049	prior_mi
52	0.0004	insulina
51	0.0039	ressonancia
50	-0.0013	renal_failure
49	-0.0004	antiplaquetario_ev
48	0.0021	procedure_type_new
47	0.0003	heart_failure
46	-0.0016	copd
45	-0.0002	admission_t0_emergency
44	-0.0005	citologia
43	0.0004	cintilografia
42	-0.0007	nyha_basal
41	0.0024	ultrassom
40	0.0028	hypertension
39	-0.0009	tomografia
38	0.0015	stroke
37	0.0005	cultura
36	-0.0002	aco
35	-0.0022	proced_invasivos_qtde
34	-0.0008	sex
33	-0.0003	diabetes
32	0.0038	cateterismo
31	0.0045	af
30	0.0088	antiarritmico
29	0.0052	reop_type_1
28	0.0087	ecocardiograma
27	0.0108	cied_final_group_1

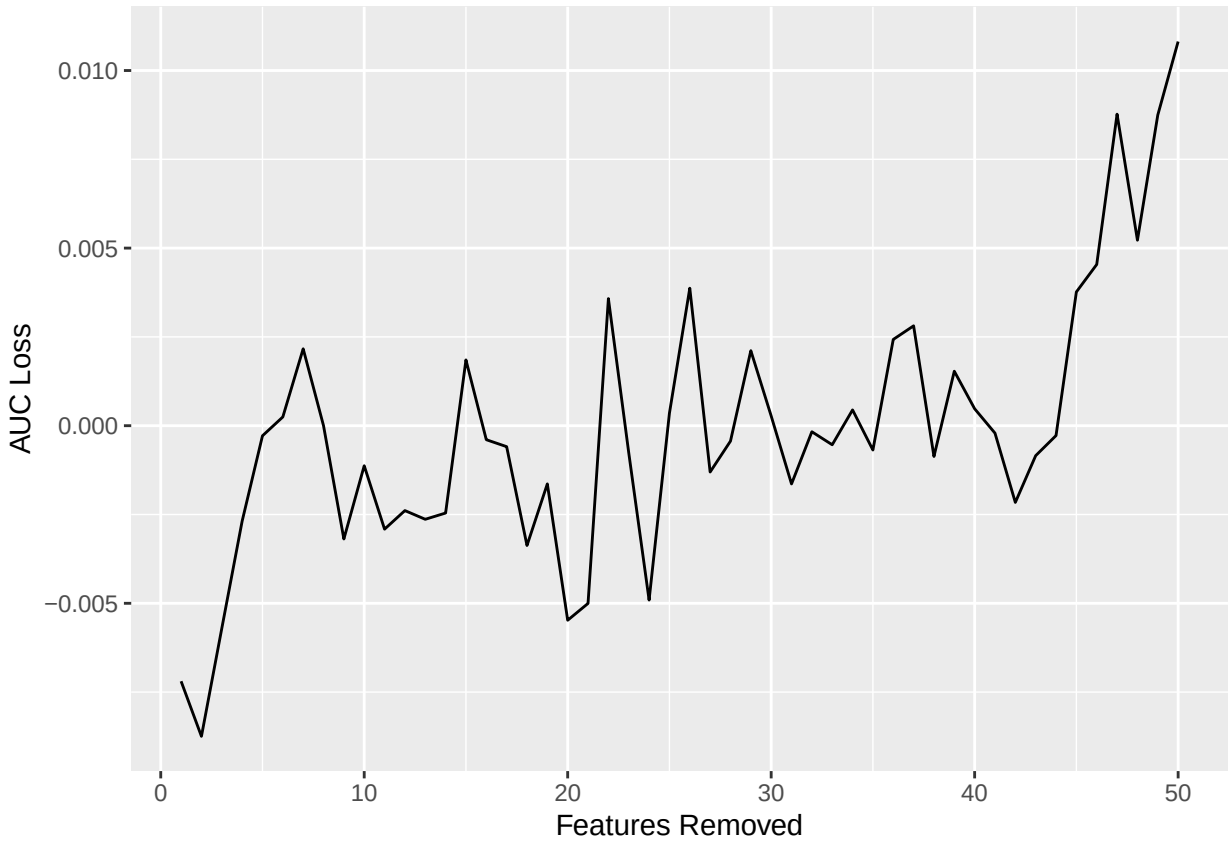
```r
if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_droped)
} else {
  selected_features <- current_features
}

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: 0.765"
```

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

```
## [1] "Trimmed Model Test AUC: 0.839"
```

```r
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:

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

1. age

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

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

Smote

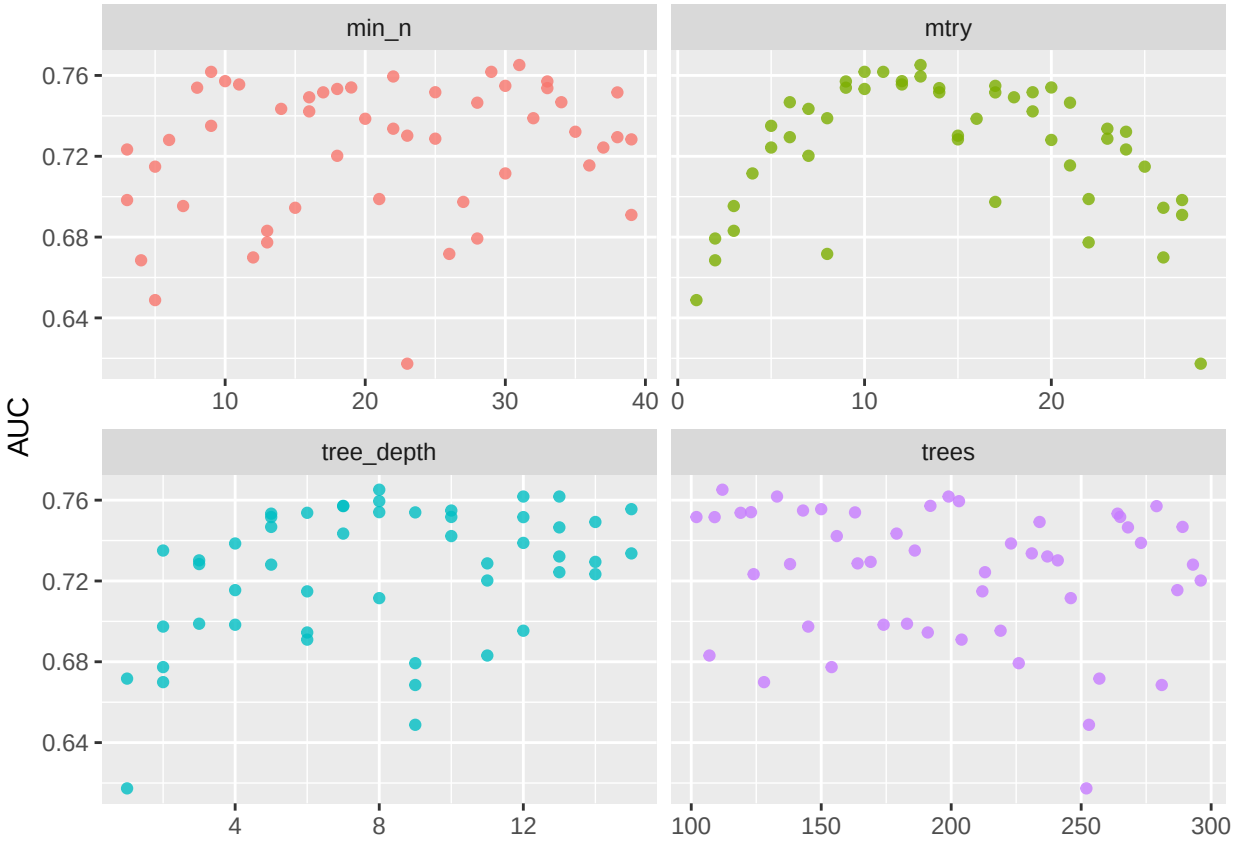
```
library(themis)

lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% select(all_of(c(selected_features, outcome_column)))) %>%
  step_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_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)
```

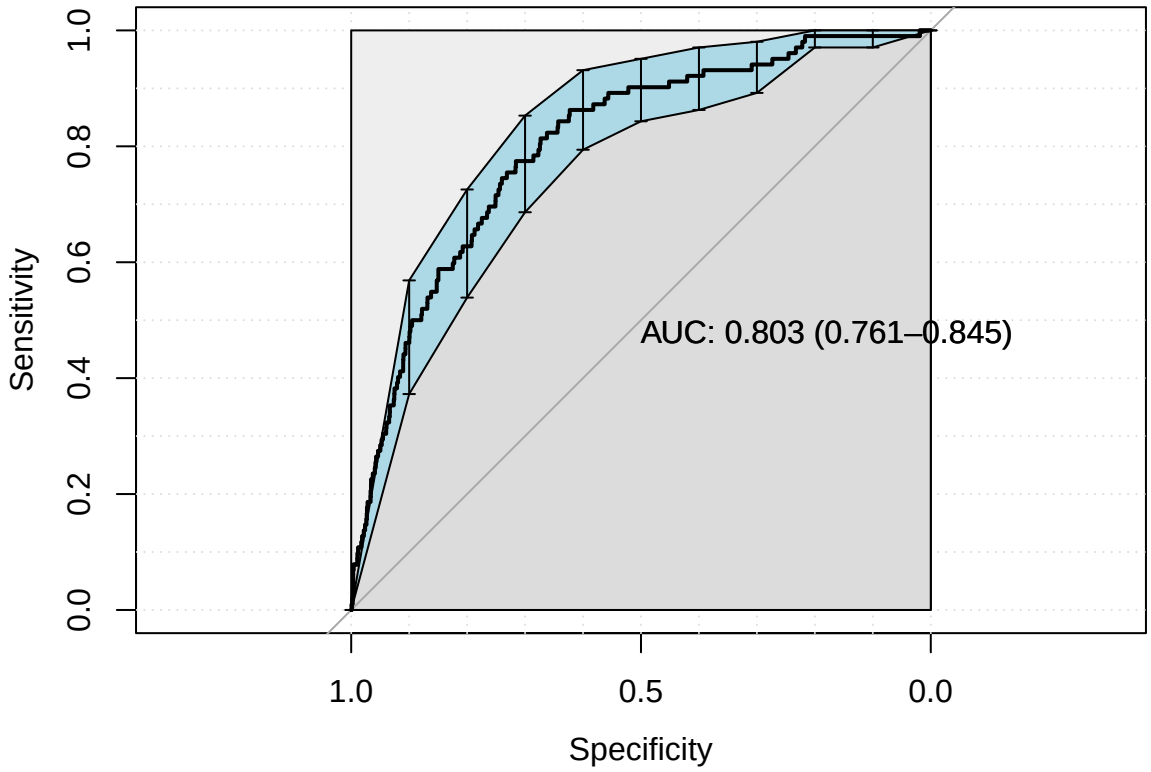
Table 2:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
13	112	31	8	0.05624	6.16728	roc_auc	binary	0.76516	5	0.01395	Preprocessor1_
10	133	29	13	0.00000	0.00000	roc_auc	binary	0.76178	5	0.02056	Preprocessor1_
11	199	9	12	0.00126	0.00001	roc_auc	binary	0.76178	5	0.01712	Preprocessor1_
13	203	22	8	0.00000	0.00000	roc_auc	binary	0.75948	5	0.01624	Preprocessor1_
12	192	10	7	0.00015	0.00013	roc_auc	binary	0.75717	5	0.01784	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 4613   97
##                       1   15    5
##
##                Accuracy : 0.9763
##                  95% CI : (0.9716, 0.9805)
##     No Information Rate : 0.9784
##     P-Value [Acc > NIR] : 0.8531
##
##                   Kappa : 0.0754
##
##  Mcnemar's Test P-Value : 1.952e-14
##
##             Sensitivity : 0.99676
##             Specificity : 0.04902
##          Pos Pred Value : 0.97941
##          Neg Pred Value : 0.25000
##              Prevalence : 0.97844
##          Detection Rate : 0.97526
##    Detection Prevalence : 0.99577
##       Balanced Accuracy : 0.52289
##
##        '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)
```

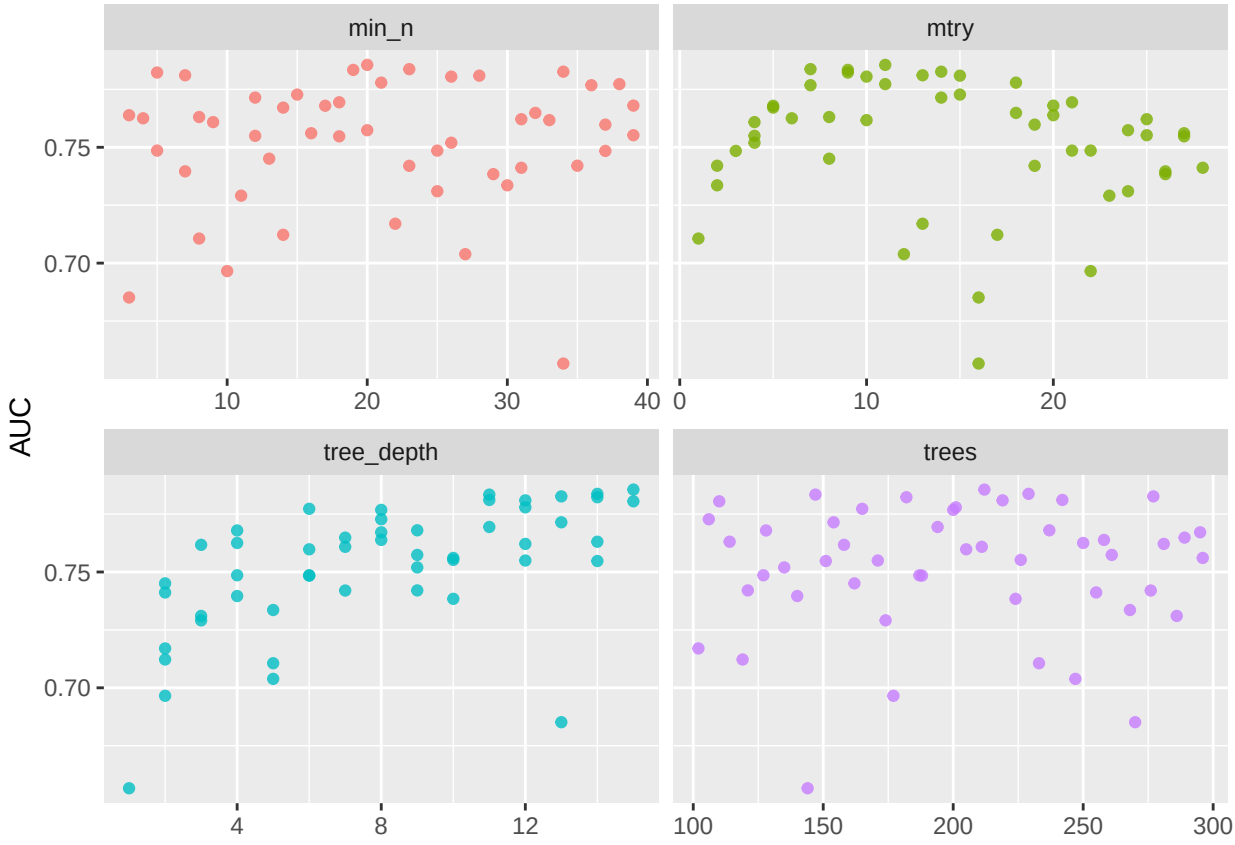
Table 3:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
11	212	20	15	0.00000	0.00000	roc_auc	binary	0.78557	5	0.01051	Preprocessor1_

Table 3: *(continued)*

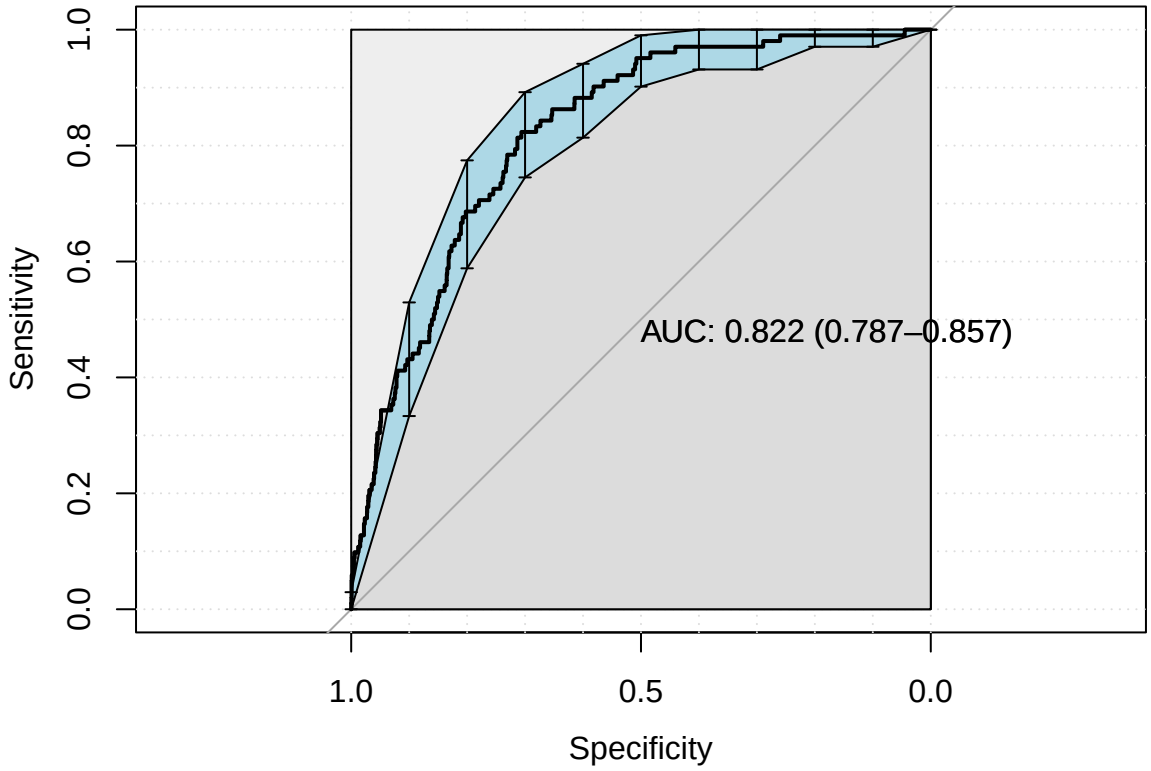
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
7	229	23	14	0.00027	0.04735	roc_auc	binary	0.78374	5	0.01383	Preprocessor1_
9	147	19	11	0.00000	0.00000	roc_auc	binary	0.78341	5	0.01219	Preprocessor1_
14	277	34	13	0.00000	0.00017	roc_auc	binary	0.78264	5	0.01437	Preprocessor1_
9	182	5	14	0.00001	0.44035	roc_auc	binary	0.78227	5	0.01414	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)
```

```r
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 3915   46
##                      1  713   56
##
##                Accuracy : 0.8395
##                  95% CI : (0.8288, 0.8499)
##     No Information Rate : 0.9784
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0941
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84594
##             Specificity : 0.54902
##          Pos Pred Value : 0.98839
##          Neg Pred Value : 0.07282
##              Prevalence : 0.97844
##          Detection Rate : 0.82770
##    Detection Prevalence : 0.83742
##       Balanced Accuracy : 0.69748
##
##        'Positive' Class : 0
##
```

```r
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)
```

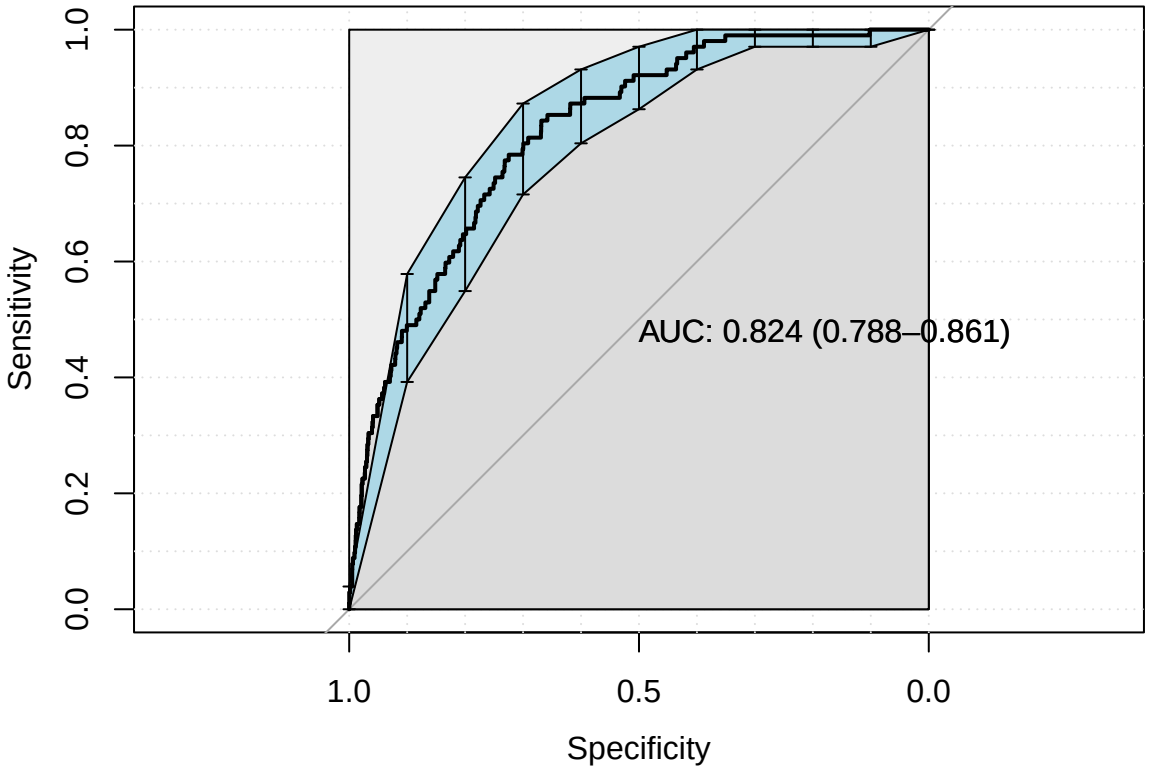
Table 4:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
9	237	20	8	0.00513	0e+00	roc_auc	binary	0.78706	5	0.01484	Preprocessor1_
16	254	11	3	0.01913	1e-05	roc_auc	binary	0.78242	5	0.01725	Preprocessor1_
11	231	30	11	0.00000	0e+00	roc_auc	binary	0.78074	5	0.01370	Preprocessor1_
11	142	29	7	0.00000	1e-05	roc_auc	binary	0.77976	5	0.01268	Preprocessor1_
11	127	32	11	0.00000	7e-05	roc_auc	binary	0.77857	5	0.01423	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)
```



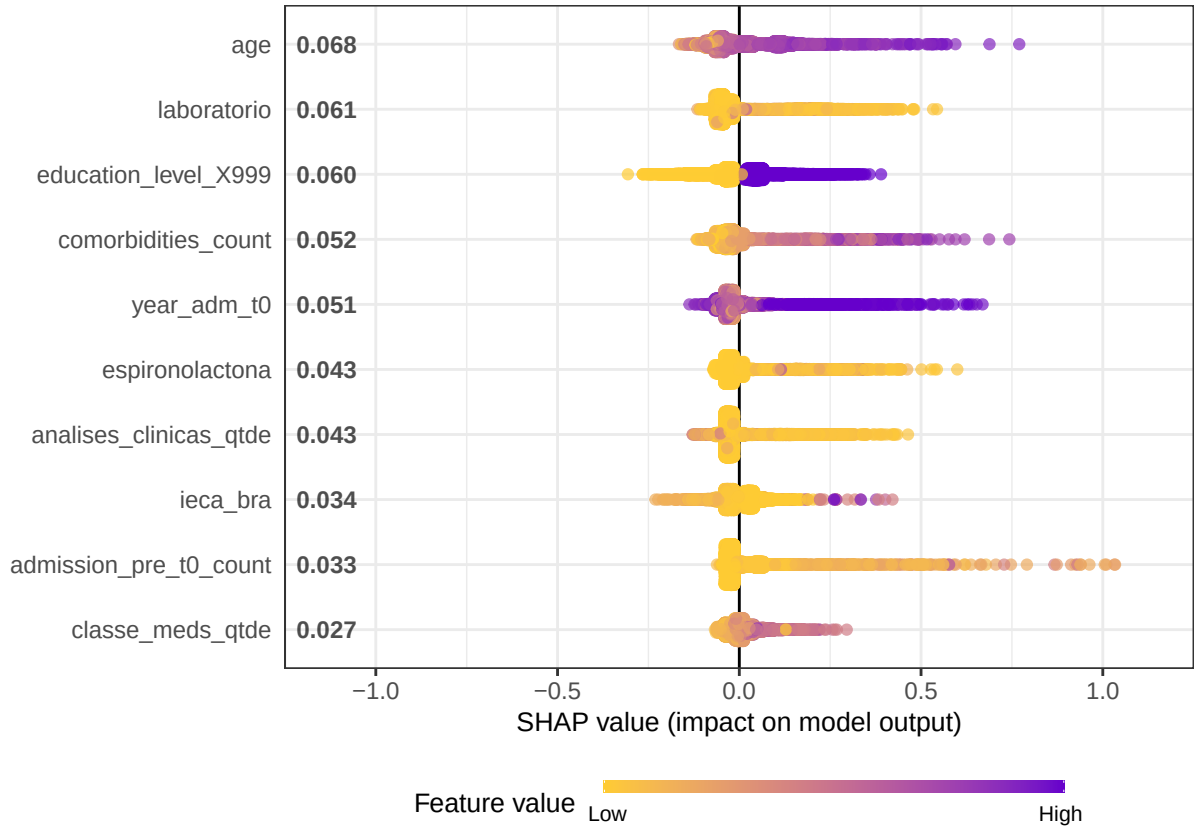

```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                      0 4628  102
##                      1    0    0
##
##                Accuracy : 0.9784
##                  95% CI : (0.9739, 0.9824)
##     No Information Rate : 0.9784
##     P-Value [Acc > NIR] : 0.5263
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9784
##          Neg Pred Value :    NaN
##              Prevalence : 0.9784
##          Detection Rate : 0.9784
##    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

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

SHAP values

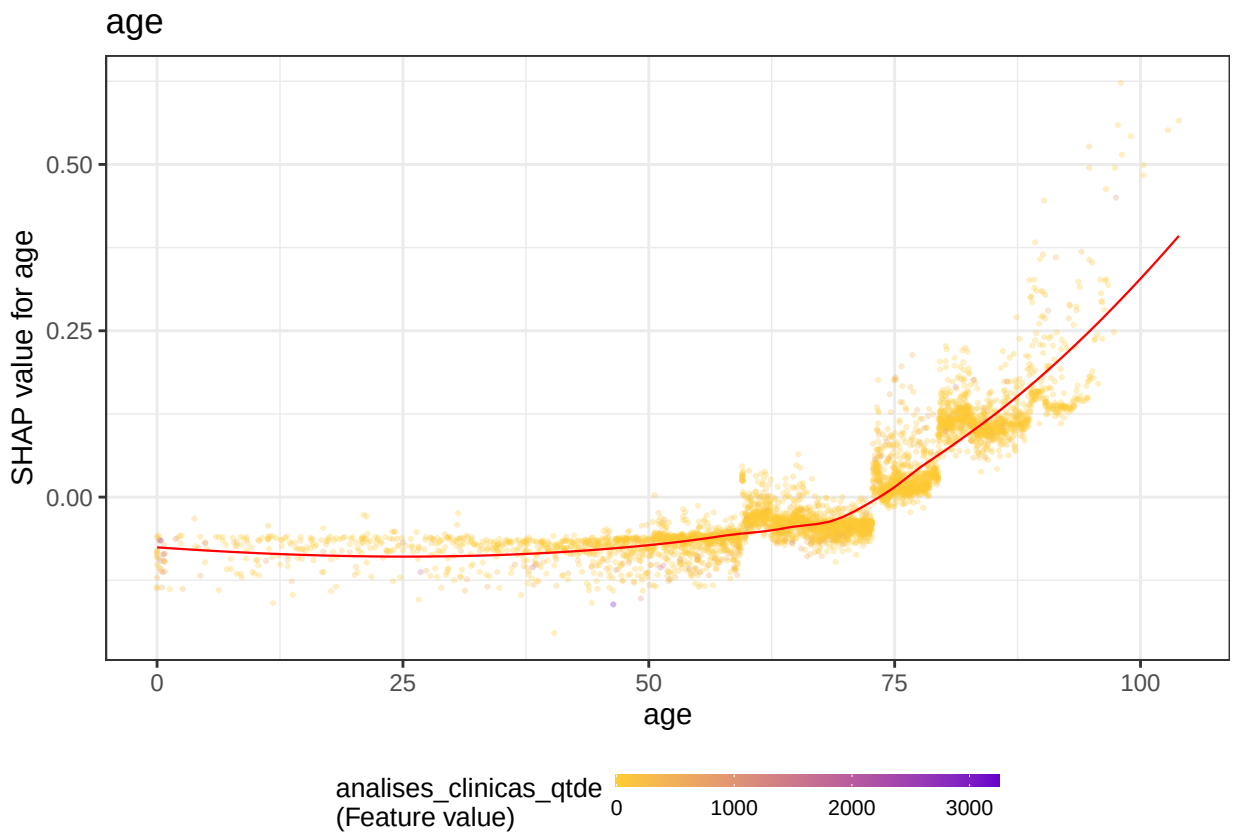
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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)
```



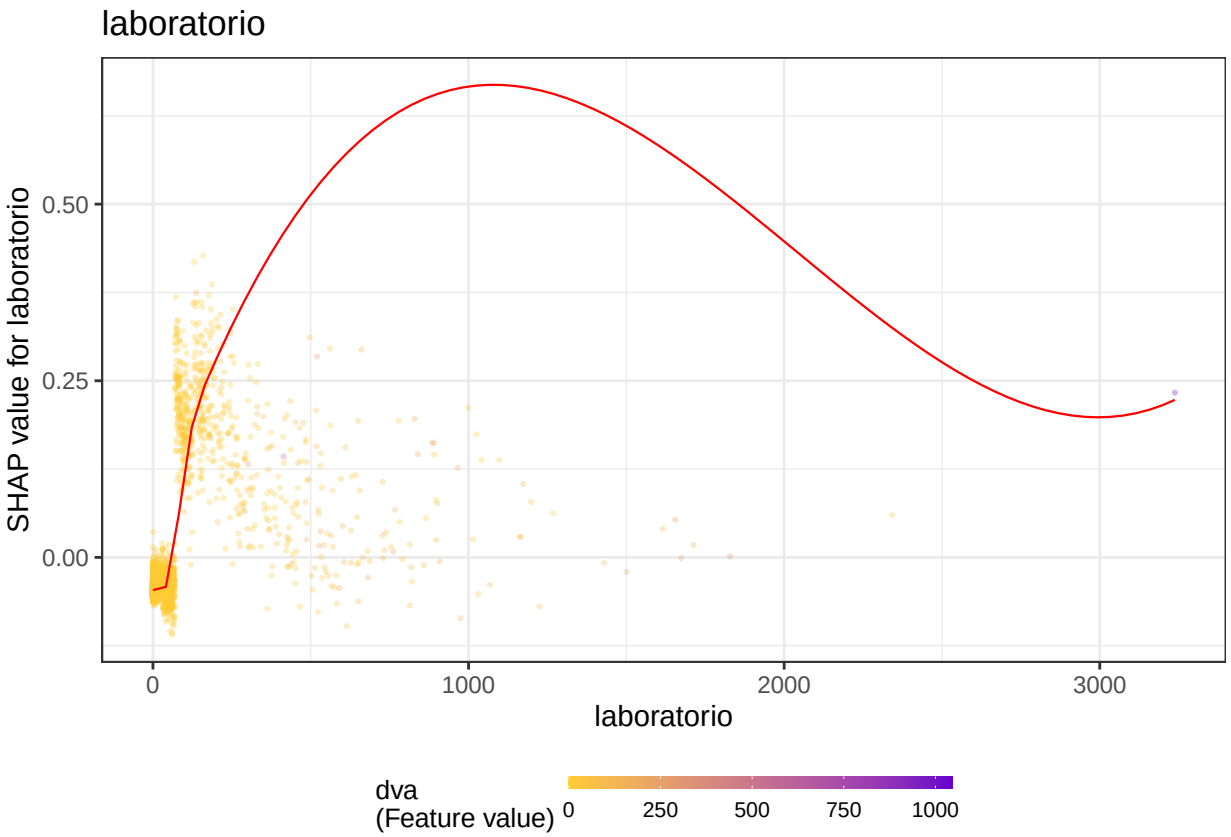

```
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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'
```




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



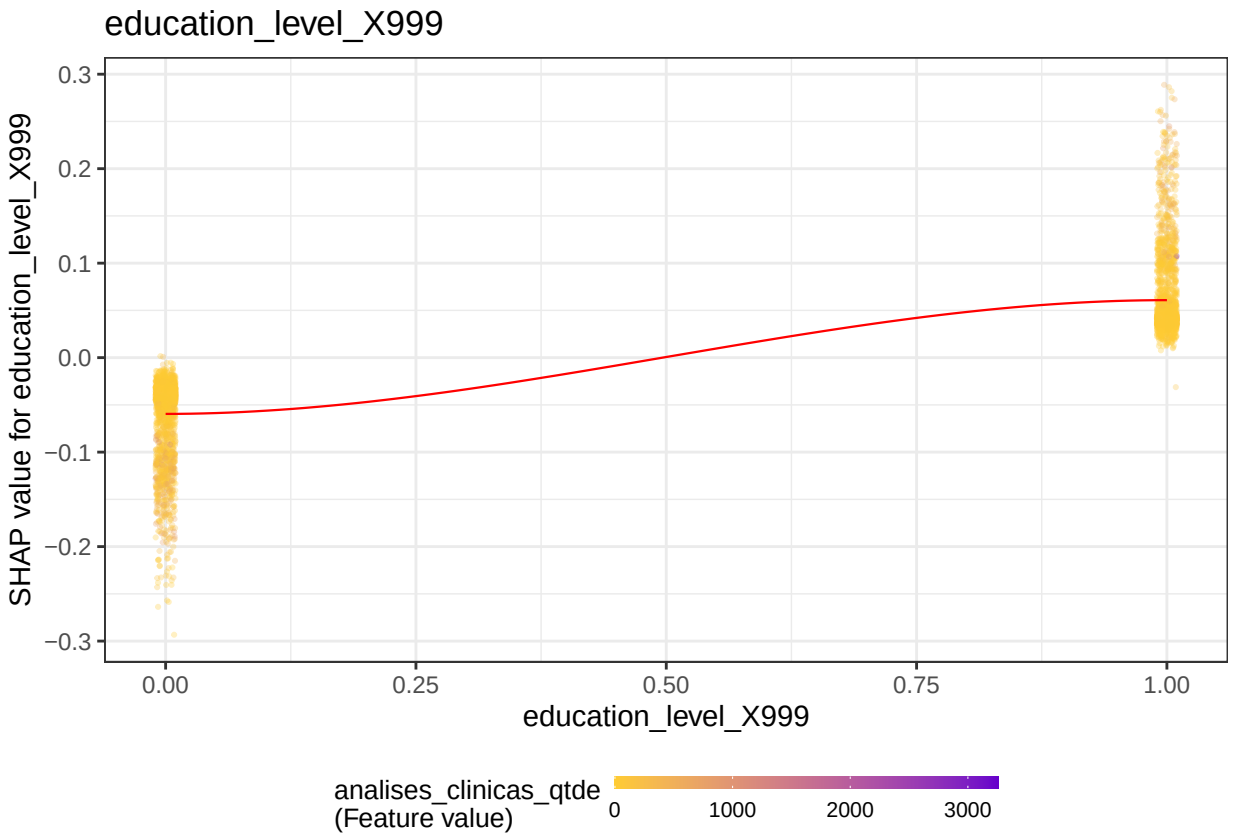

```
## `geom_smooth()` using formula 'y ~ x'
```

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

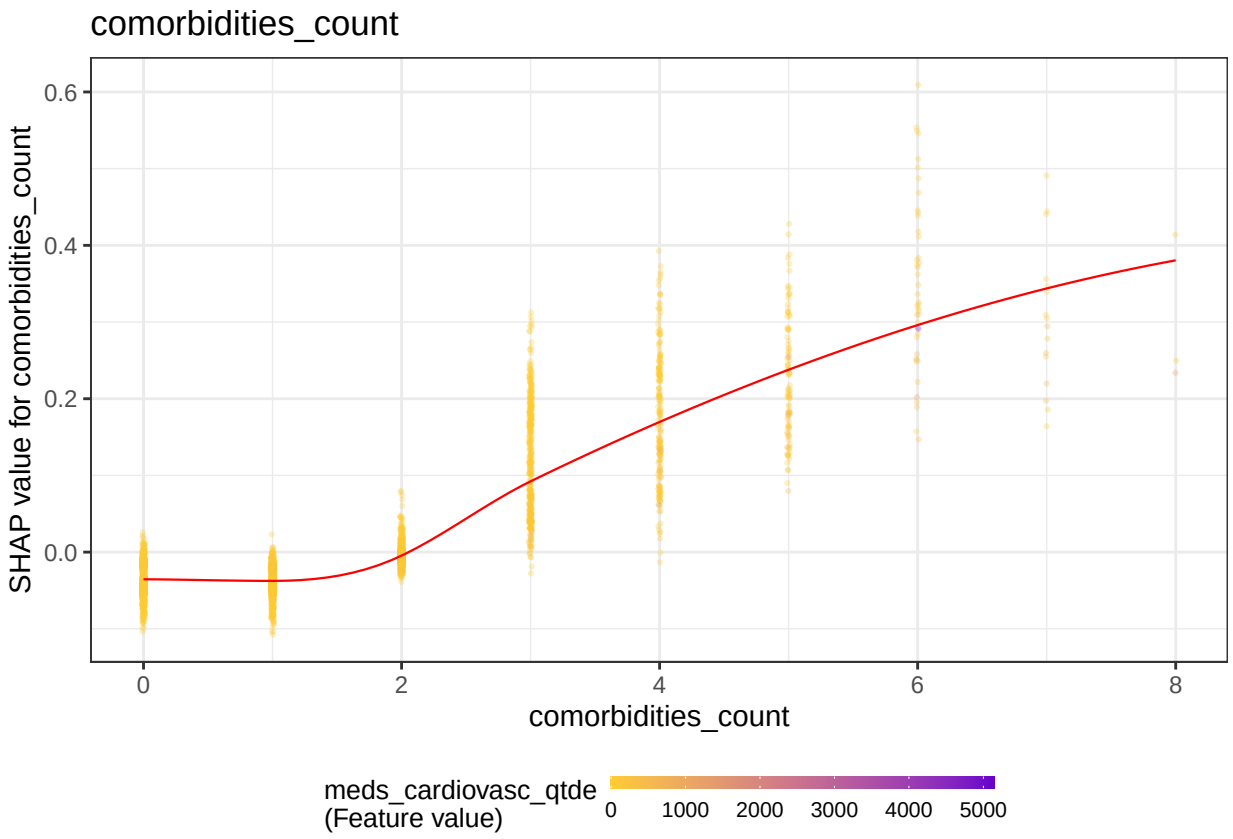
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 1.1613e-28
```

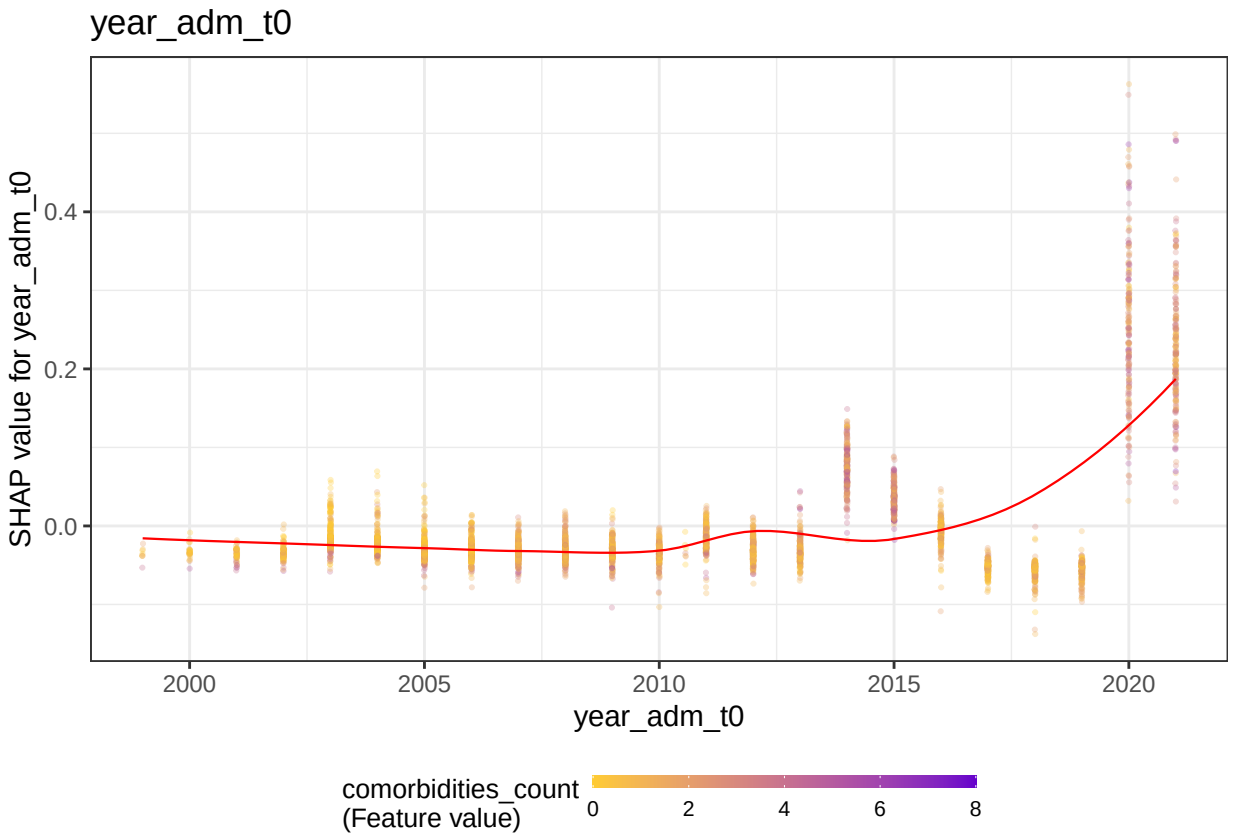
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are
## other near singularities as well. 1.01
```




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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -0.04

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 2.04

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 4.8353e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are
## other near singularities as well. 4
```

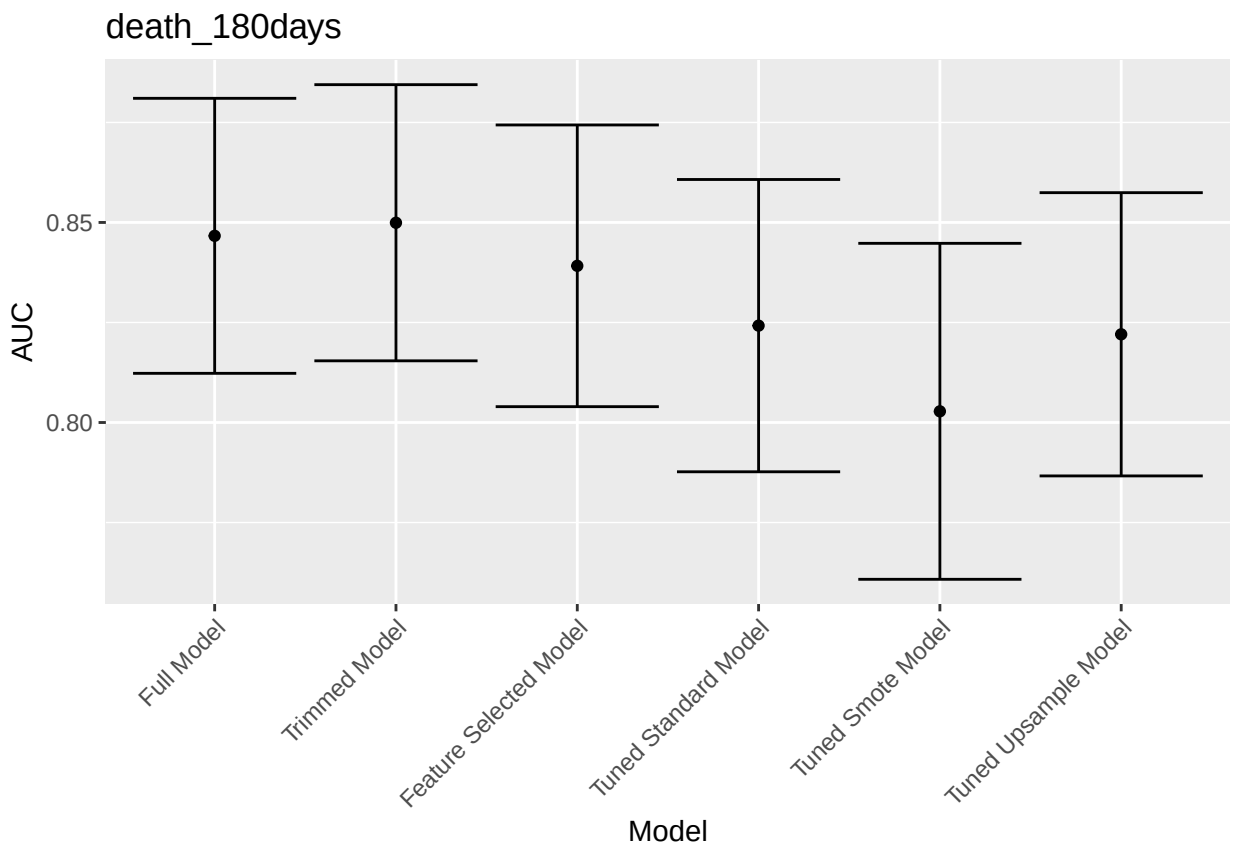



```
## `geom_smooth()` using formula 'y ~ x'
```



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 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_

) %>%
  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))
```