

# Final Model - death\_3year

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. race
4. education\_level
5. underlying\_heart\_disease
6. heart\_disease
7. nyha\_basal
8. hypertension
9. prior\_mi
10. heart\_failure
11. af
12. cardiac\_arrest
13. valvopathy
14. diabetes
15. renal\_failure
16. hemodialysis
17. stroke
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. year\_adm\_t0
28. icu\_t0
29. dialysis\_t0
30. admission\_t0\_emergency
31. aco
32. antiaritmico
33. ieca\_bra
34. dva
35. digoxina
36. estatina
37. diuretico

38. vasodilatador  
39. insuf\_cardiaca  
40. espironolactona  
41. antiplaquetario\_ev  
42. insulina  
43. anticonvulsivante  
44. psicofarmacos  
45. antifungico  
46. classe\_meds\_qtde  
47. meds\_cardiovasc\_qtde  
48. meds\_antimicrobianos  
49. outros\_proced\_cirurgicos  
50. icp  
51. angioplastia  
52. cateterismo  
53. eletrofisiologia  
54. cateter\_venoso\_central  
55. proced\_invasivos\_qtde  
56. transfusao  
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. angiografia  
70. cintilografia  
71. ecocardiograma  
72. endoscopia  
73. flebografia  
74. pet\_ct  
75. ultrassom  
76. tomografia  
77. radiografia  
78. ressonancia  
79. exames\_imagem\_qtde  
80. 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: 0.799"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.801"

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

unimportant_features %>%
  gluedown::md_order()

1. heart_disease
2. angioplastia

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: 0.796"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.802"

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
80	0.0000	angiografia
78	0.0035	angiografia
77	0.0007	holter
76	0.0024	procedure_type_new
75	0.0007	histopatologia_qtde
74	0.0007	tilt_teste
73	0.0018	stroke
72	0.0032	cintilografia
71	0.0022	eletrofisiologia
70	0.0006	cateter_venoso_central
69	0.0006	angio_tc
68	0.0005	teste_esforco
67	0.0006	copd
66	0.0004	transfusao
65	0.0009	antiplaquetario_ev
64	0.0002	dialysis_t0
63	0.0013	hemodialysis
62	0.0019	cateterismo
61	0.0002	antifungico
60	0.0015	endoscopia
59	0.0016	cardiac_arrest
58	0.0021	citologia
57	0.0022	ressonancia
56	0.0010	flebografia
55	0.0021	pet_ct
54	0.0018	icp
53	0.0033	insulina
52	0.0019	reop_type_1
51	0.0021	prior_mi
50	0.0007	admission_t0_emergency
49	0.0001	cultura
48	-0.0011	heart_failure
47	-0.0007	aco
46	0.0015	af
45	-0.0004	ecocardiograma
44	-0.0007	outros_proced_cirurgicos
43	0.0019	hypertension
42	-0.0002	renal_failure
41	0.0025	admission_pre_t0_180d
40	0.0005	sex
39	0.0013	tomografia
38	0.0022	analises_clinicas_qtde
37	0.0007	diabetes
36	0.0027	underlying_heart_disease
35	0.0021	digoxina
34	0.0012	bic
33	0.0012	ultrassom
32	0.0004	radiografia

Table 1: (*continued*)

Number of Features	AUC Loss	Least Important Feature
31	0.0029	anticonvulsivante
30	0.0030	proced_invasivos_qtde
29	0.0024	race
28	0.0026	valvopathy
27	0.0044	cied_final_group_1
26	0.0031	dva
25	0.0041	procedure_type_1
24	0.0025	cied_final_1
23	0.0061	antiarritmico
22	0.0079	metodos_graficos_qtde
21	0.0094	estatina
20	0.0081	nyha_basal
19	0.0239	examens_imagem_qtde

```

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)

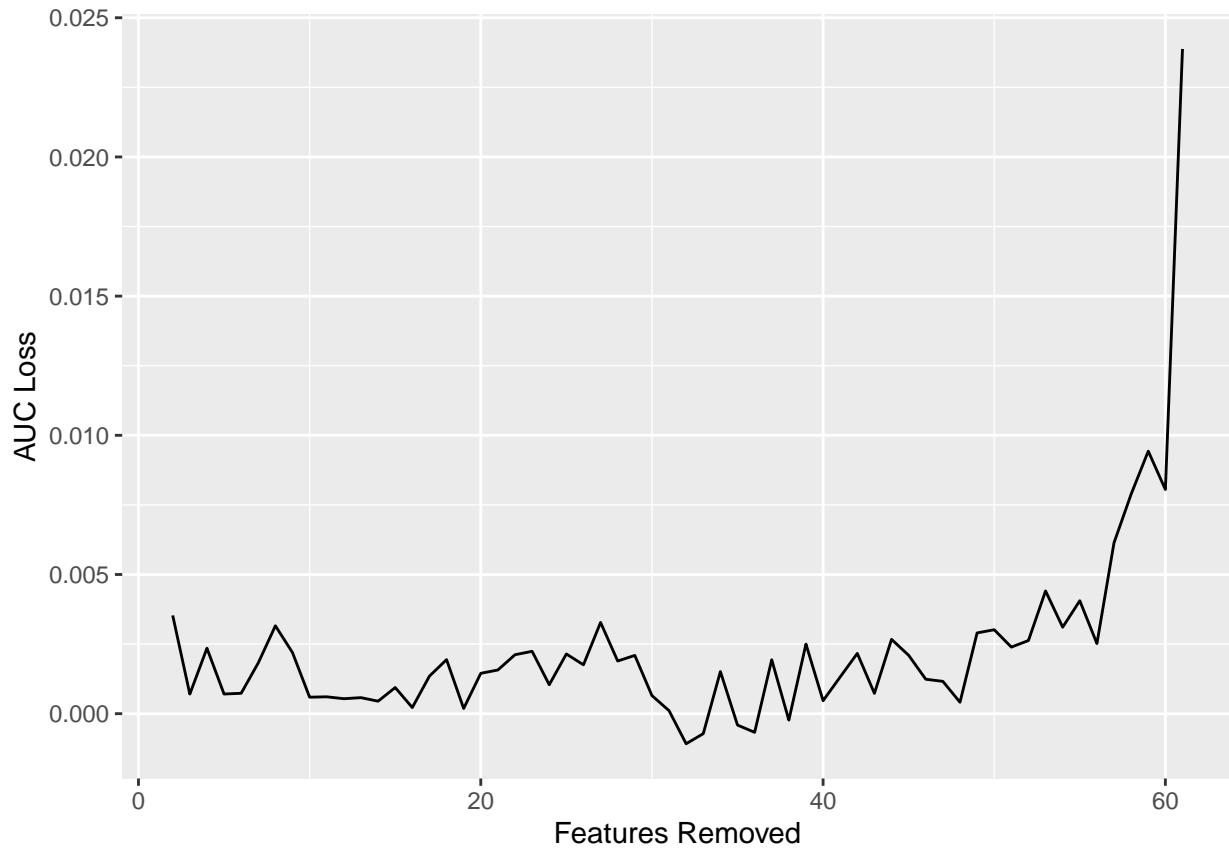
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.775"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.781"

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. year\_adm\_t0
3. admission\_pre\_t0\_count
4. classe\_meds\_qtde
5. espironolactona
6. comorbidities\_count
7. education\_level
8. meds\_cardiovasc\_qtde
9. ieca\_bra
10. icu\_t0
11. vasodilatador
12. insuf\_cardiaca
13. laboratorio
14. equipe\_multiprof
15. diuretico
16. psicofarmacos
17. ecg
18. exames\_imagem\_qtde
19. meds\_antimicrobianos

```
# 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	241	27	7	0.03267	0.00006	roc_auc	binary	0.77197	5	0.00313	Preprocessor1
19	211	5	13	0.05509	4.87284	roc_auc	binary	0.77012	5	0.00302	Preprocessor1
7	237	4	4	0.07638	0.00095	roc_auc	binary	0.76806	5	0.00525	Preprocessor1
16	107	38	13	0.01635	0.00000	roc_auc	binary	0.75970	5	0.00362	Preprocessor1
12	179	22	15	0.00640	8.74150	roc_auc	binary	0.75637	5	0.00623	Preprocessor1

```

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

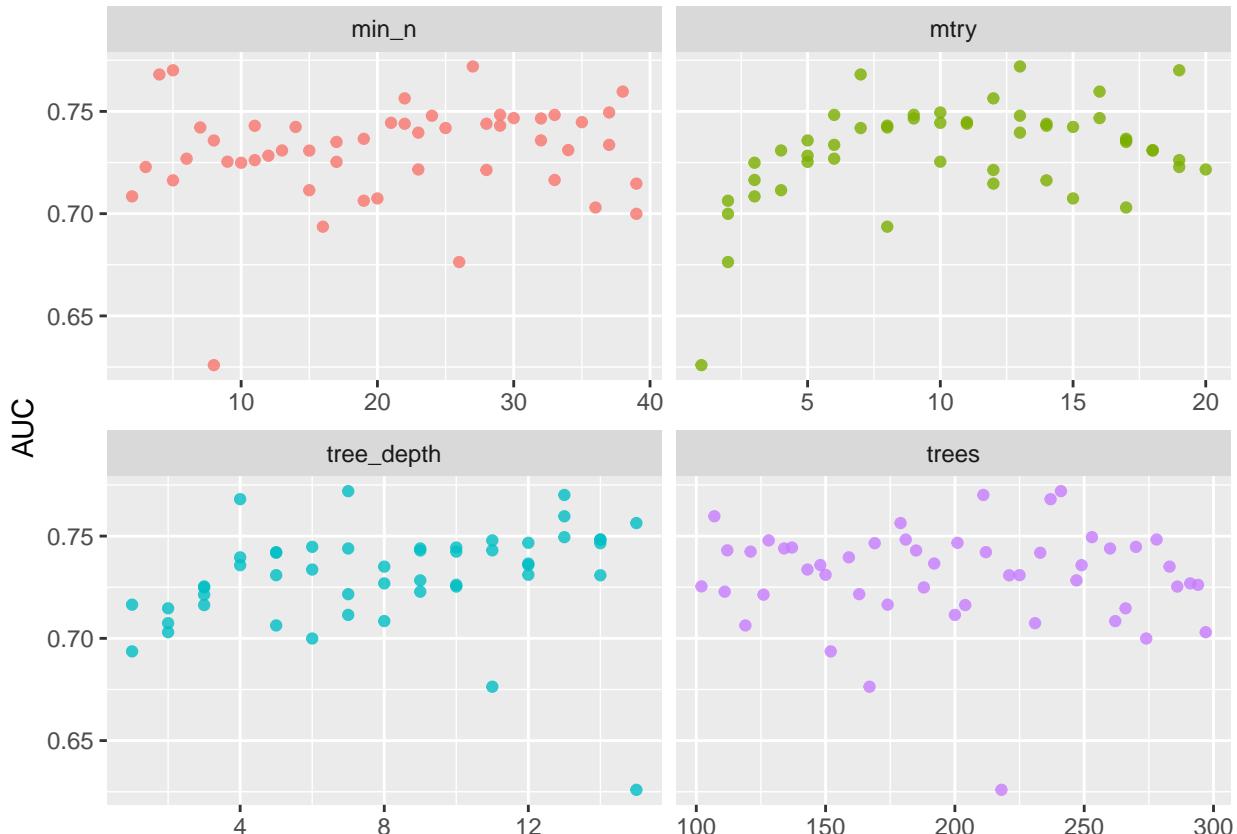
lightgbm_tune %>%
  collect_metrics() %>%

```

```

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")

```



```

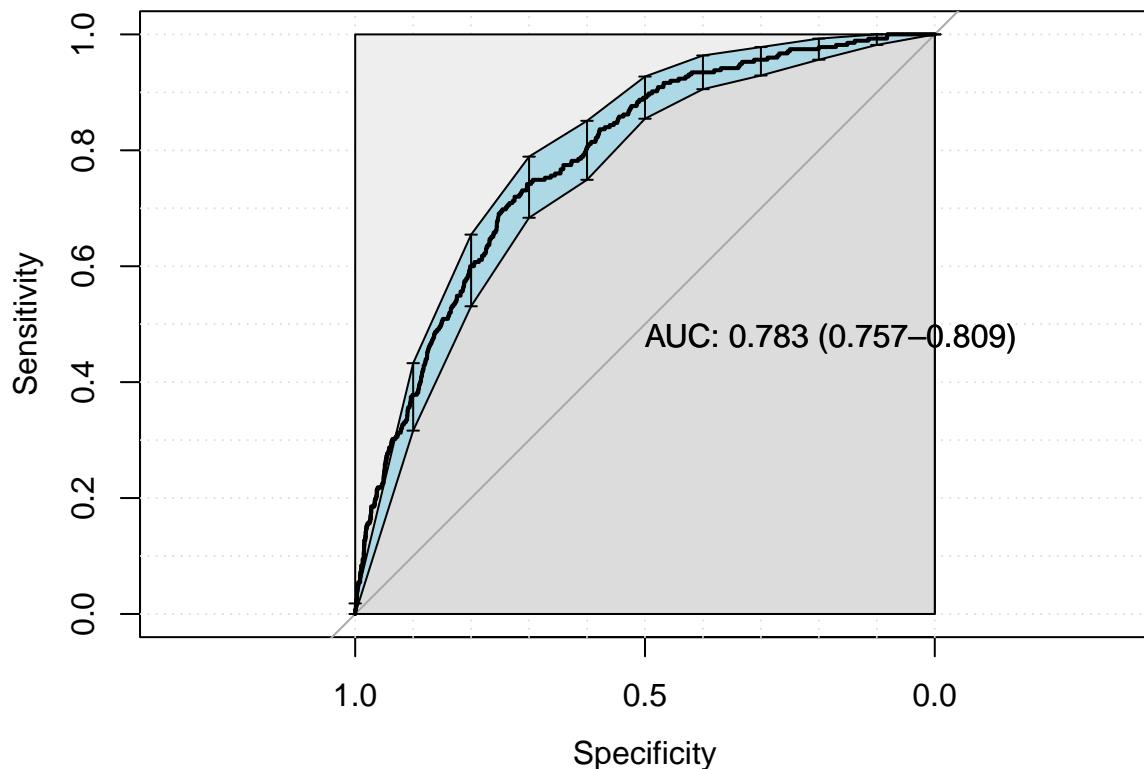
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4411  255
##                      1    44   20
##
##          Accuracy : 0.9368
## 95% CI : (0.9295, 0.9436)
##  No Information Rate : 0.9419
## P-Value [Acc > NIR] : 0.9346
##
##          Kappa : 0.0982
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.99012
##          Specificity  : 0.07273
##  Pos Pred Value  : 0.94535
##  Neg Pred Value  : 0.31250
##          Prevalence  : 0.94186
##  Detection Rate  : 0.93256
## Detection Prevalence : 0.98647
##  Balanced Accuracy : 0.53143
##
##  '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:

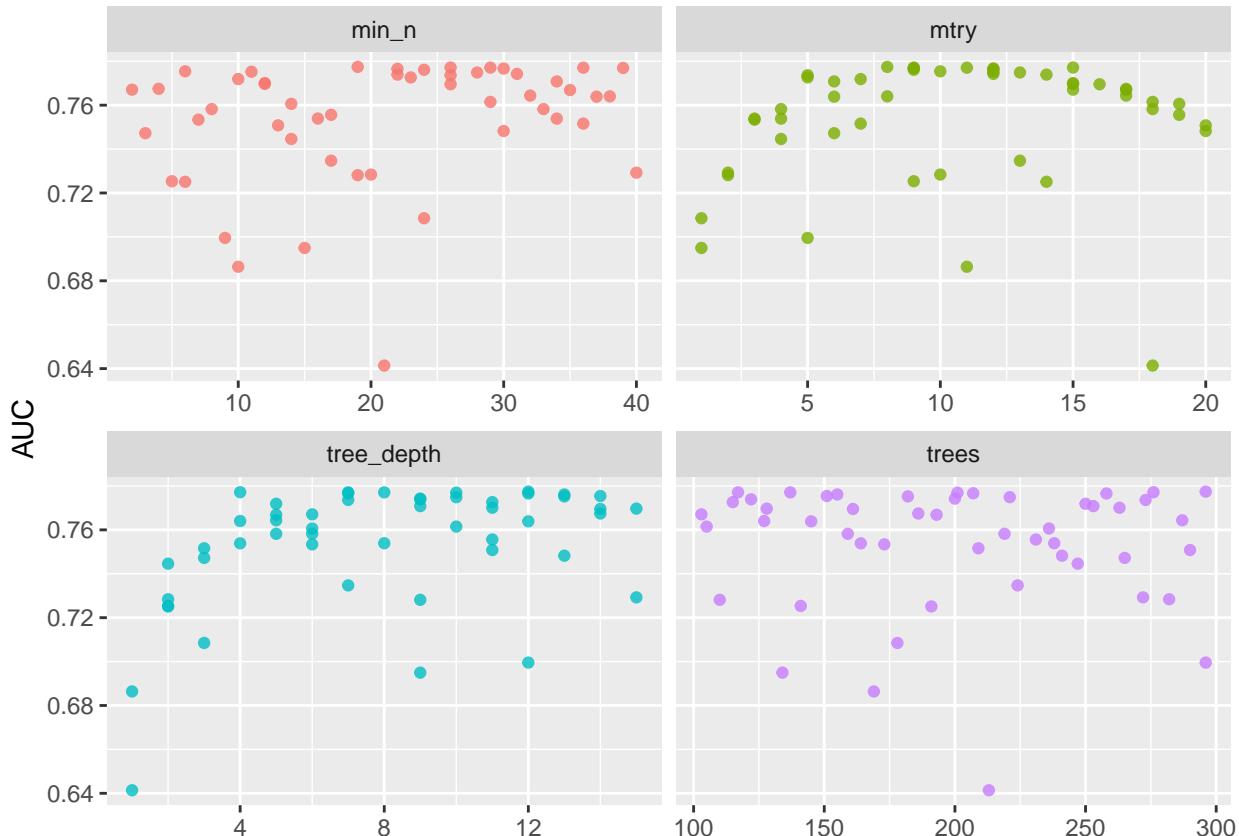
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
8	296	19	12	0.00000	0.00002	roc_auc	binary	0.77743	5	0.00753	Preprocessor1

Table 3: (continued)

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
15	276	26	4	0.00986	4.96072	roc_auc	binary	0.77715	5	0.00492	Preprocessor1
9	117	29	7	0.00001	1.92407	roc_auc	binary	0.77710	5	0.00624	Preprocessor1
11	137	36	8	0.00000	0.00013	roc_auc	binary	0.77707	5	0.00641	Preprocessor1
9	201	39	10	0.00000	1.20045	roc_auc	binary	0.77695	5	0.00762	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)

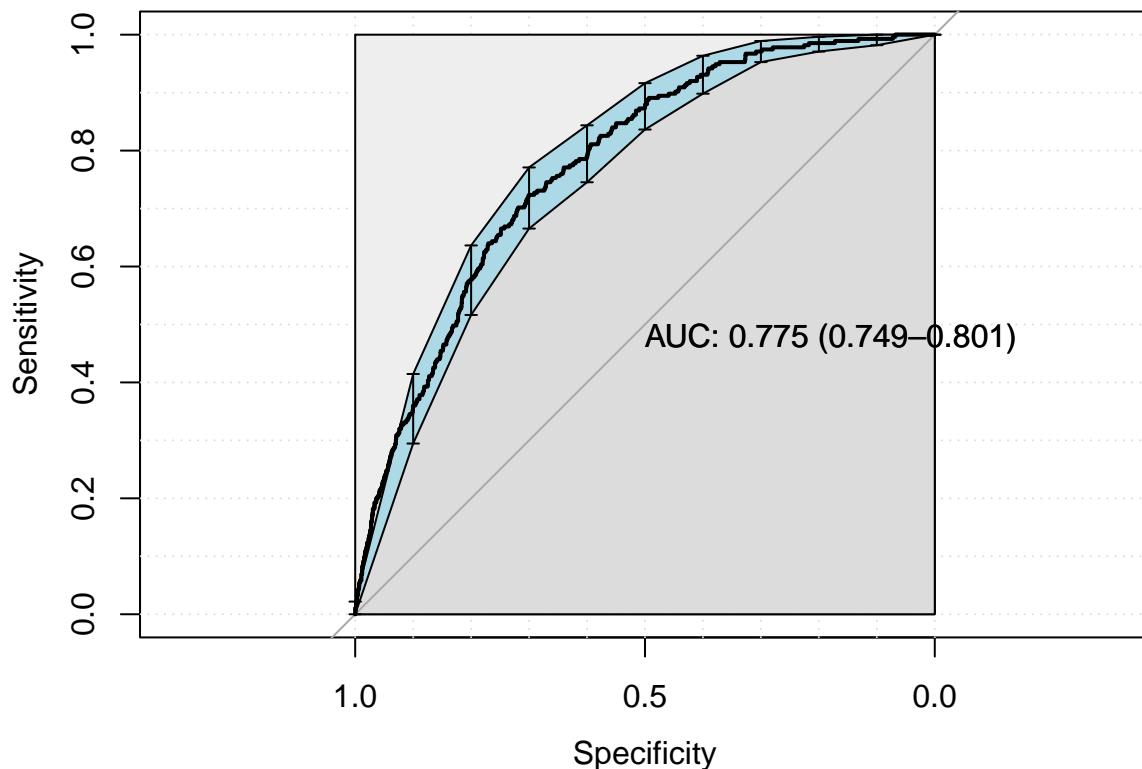
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)
```

```

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 3465 104
##                      1 990 171
##
##          Accuracy : 0.7687
##          95% CI : (0.7564, 0.7807)
##  No Information Rate : 0.9419
##  P-Value [Acc > NIR] : 1
##
##          Kappa : 0.1591
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.7778
##          Specificity : 0.6218
##  Pos Pred Value : 0.9709
##  Neg Pred Value : 0.1473
##          Prevalence : 0.9419
##  Detection Rate : 0.7326
##  Detection Prevalence : 0.7545
##          Balanced Accuracy : 0.6998
##
##  '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)

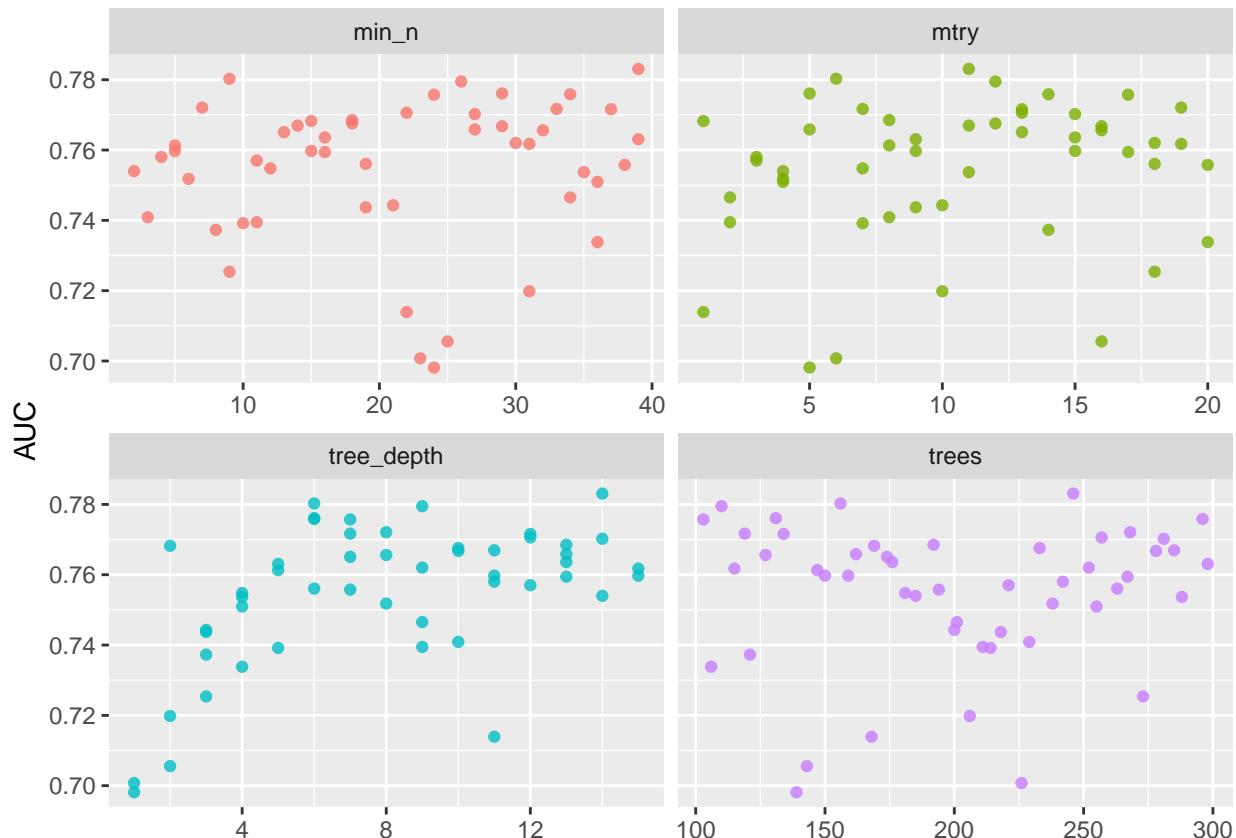
```

Table 4:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
11	246	39	14	0.00629	0.00000	roc_auc	binary	0.78309	5	0.00759	Preprocessor1
6	156	9	6	0.01362	0.00000	roc_auc	binary	0.78027	5	0.00655	Preprocessor1
12	110	26	9	0.04612	0.00013	roc_auc	binary	0.77950	5	0.00729	Preprocessor1
5	131	29	6	0.00876	0.00000	roc_auc	binary	0.77609	5	0.00759	Preprocessor1
14	296	34	6	0.00324	0.30856	roc_auc	binary	0.77586	5	0.00740	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")
```

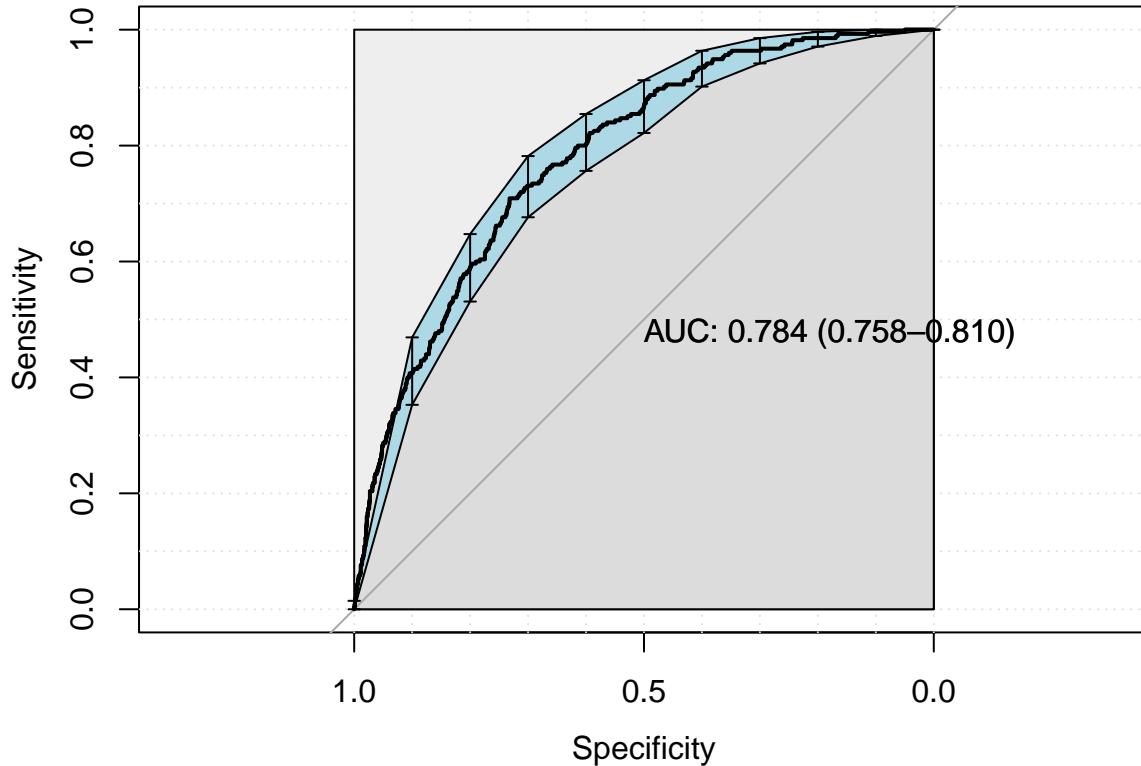


```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4455  275
##                      1     0     0
##
##          Accuracy : 0.9419
##          95% CI : (0.9348, 0.9484)
##  No Information Rate : 0.9419
##  P-Value [Acc > NIR] : 0.516
##
##          Kappa : 0
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.0000
##          Specificity : 0.0000
##  Pos Pred Value : 0.9419
##  Neg Pred Value :      NaN
##          Prevalence : 0.9419
##  Detection Rate : 0.9419
## 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)

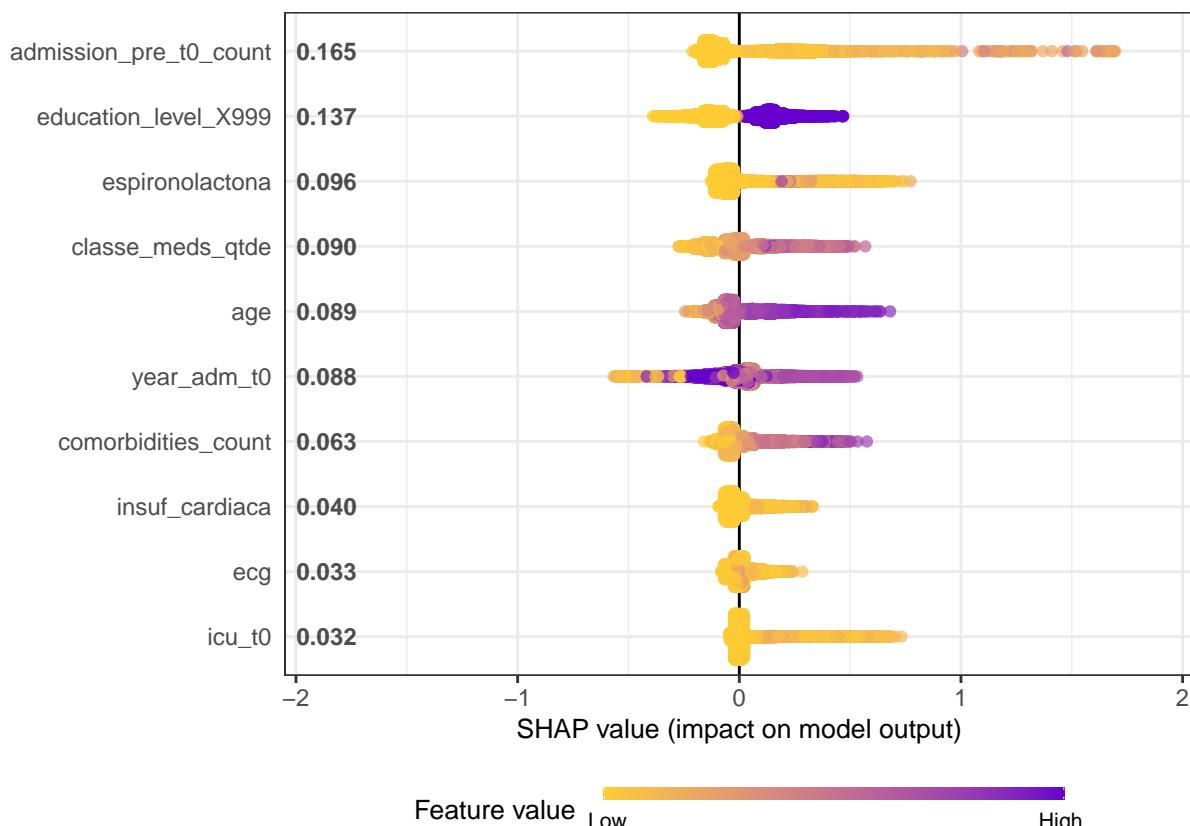
trained_rec <- prep(lightgbm_recipe, training = df_train)

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'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -0.125

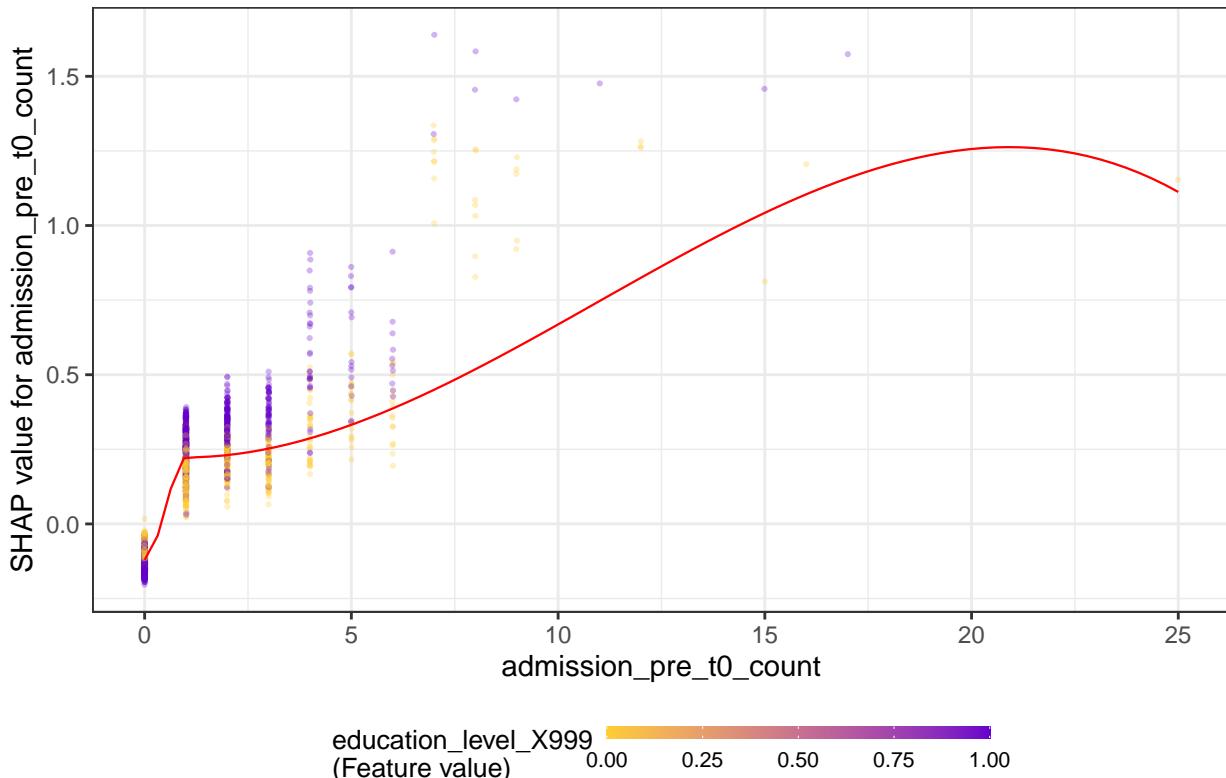
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.125

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 1.3776e-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'

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

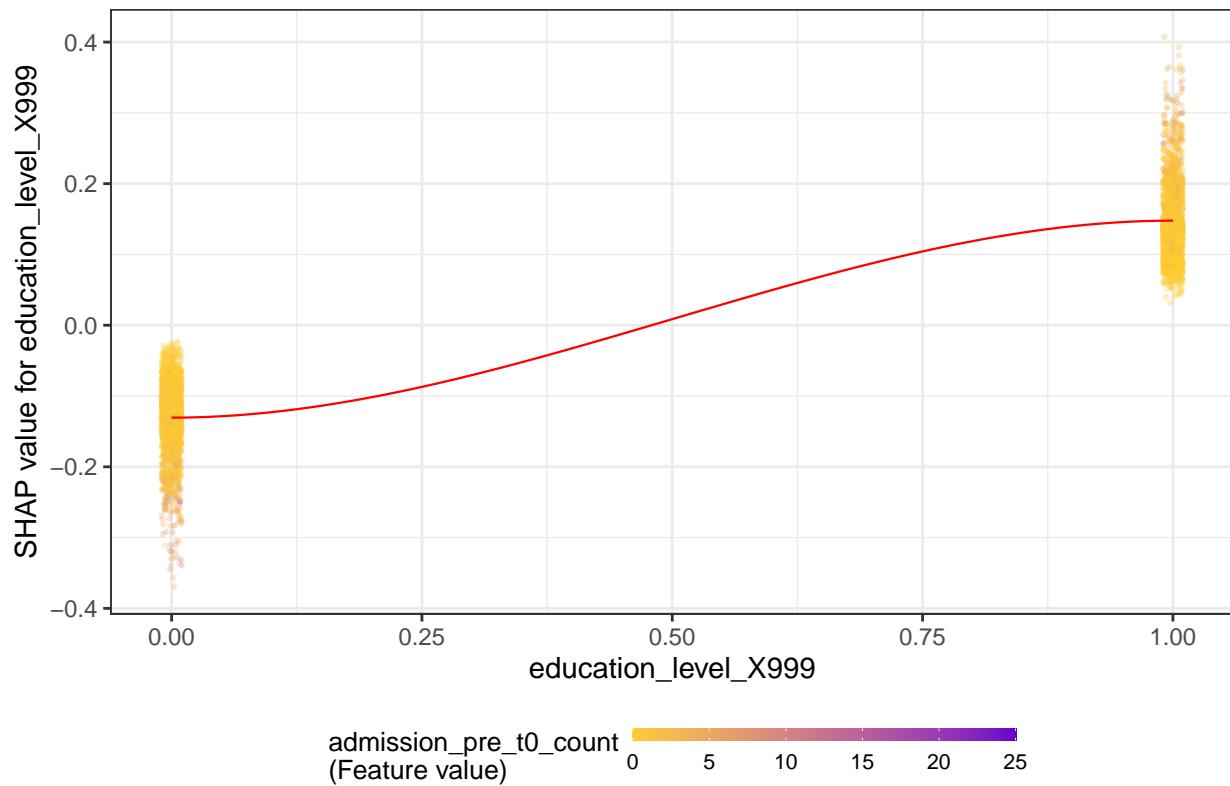
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 2.1666e-28

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

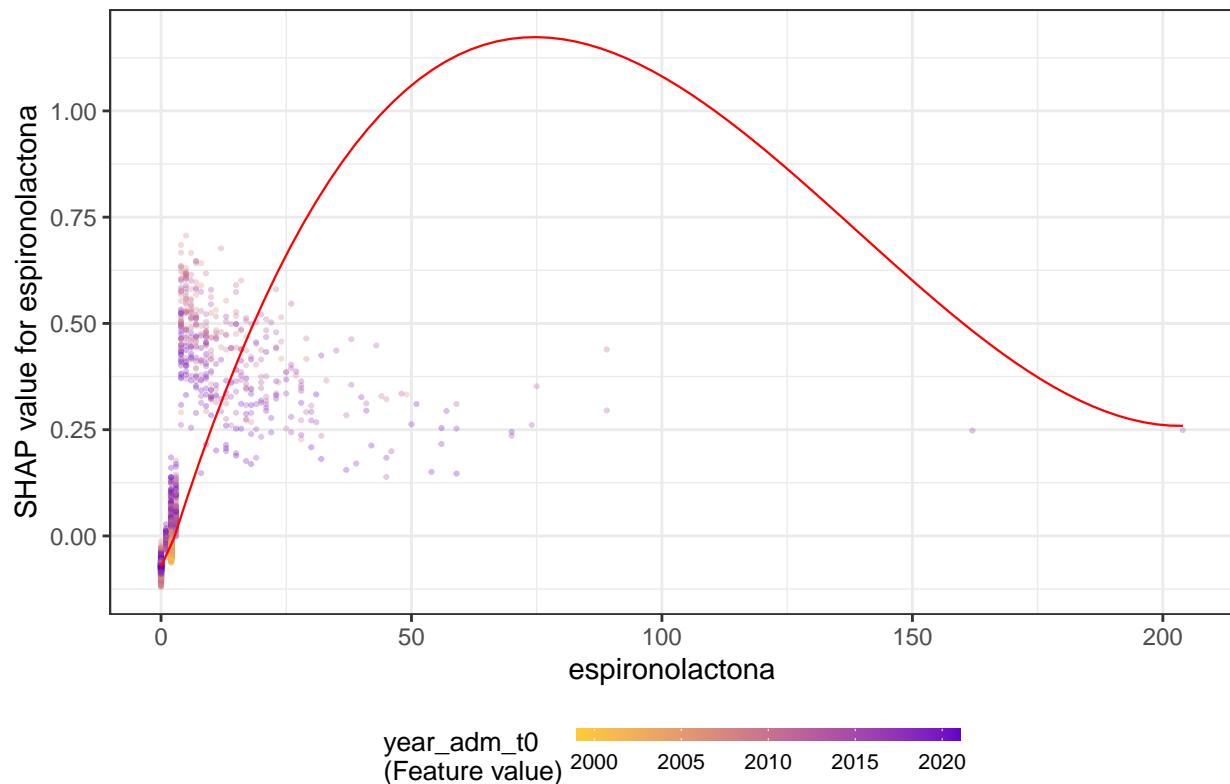
```

education\_level\_X999

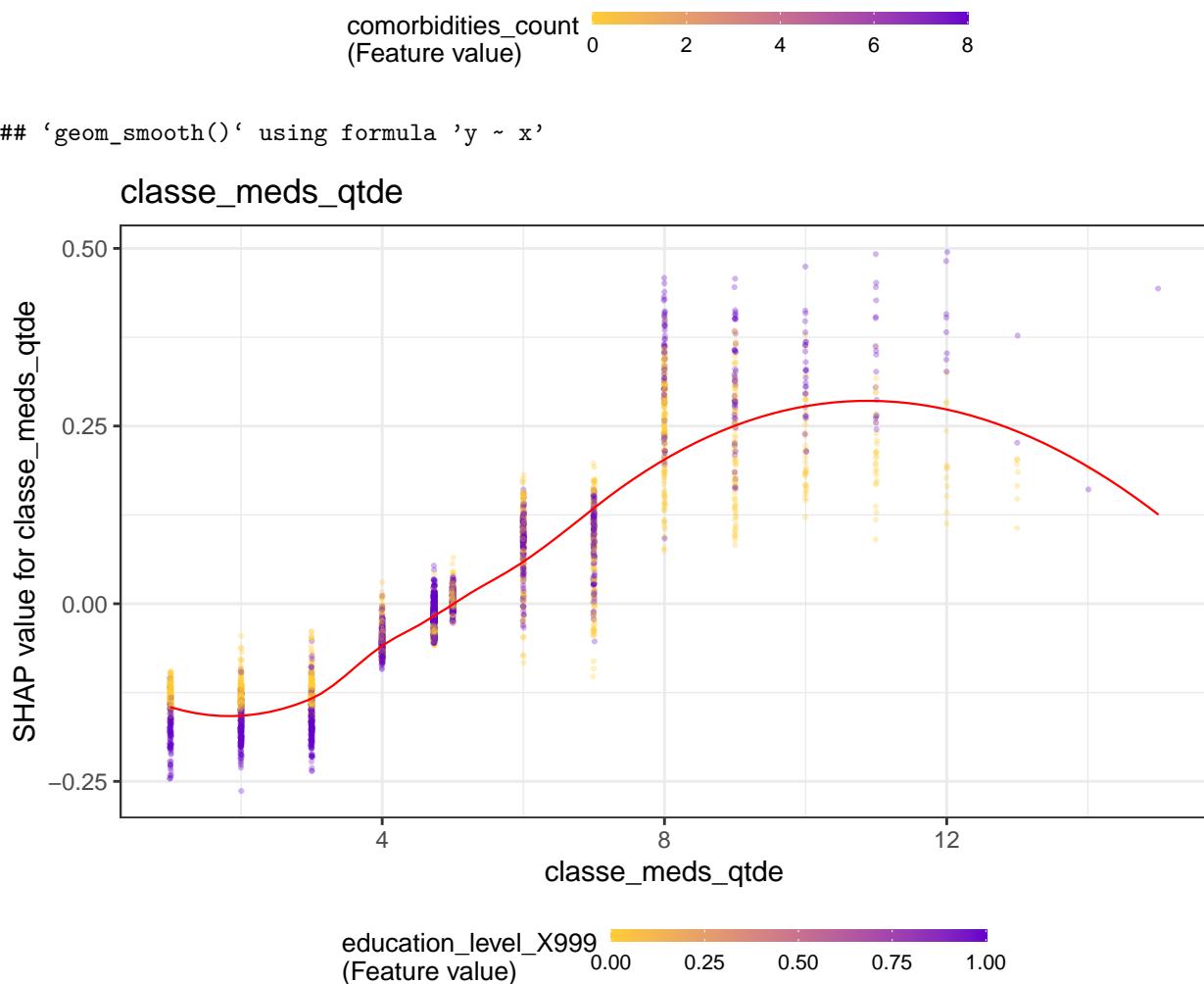
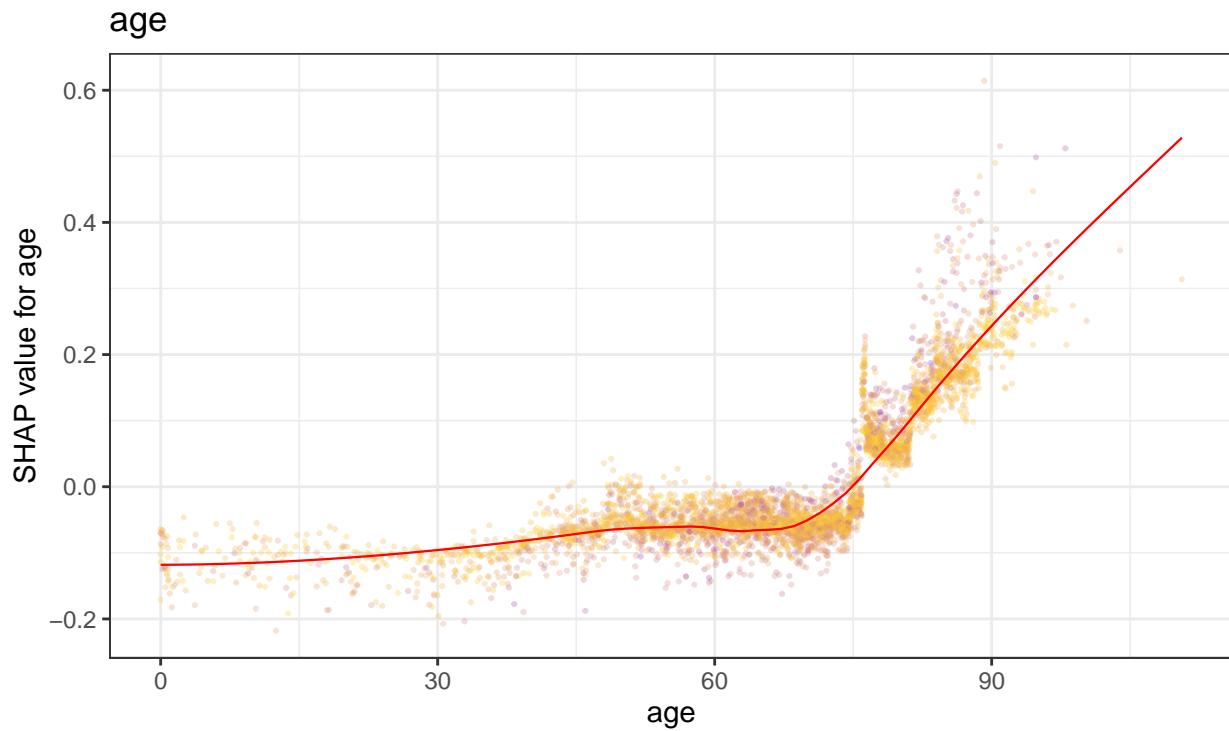


```
## `geom_smooth()` using formula 'y ~ x'
```

espironolactona



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



## Models Comparison

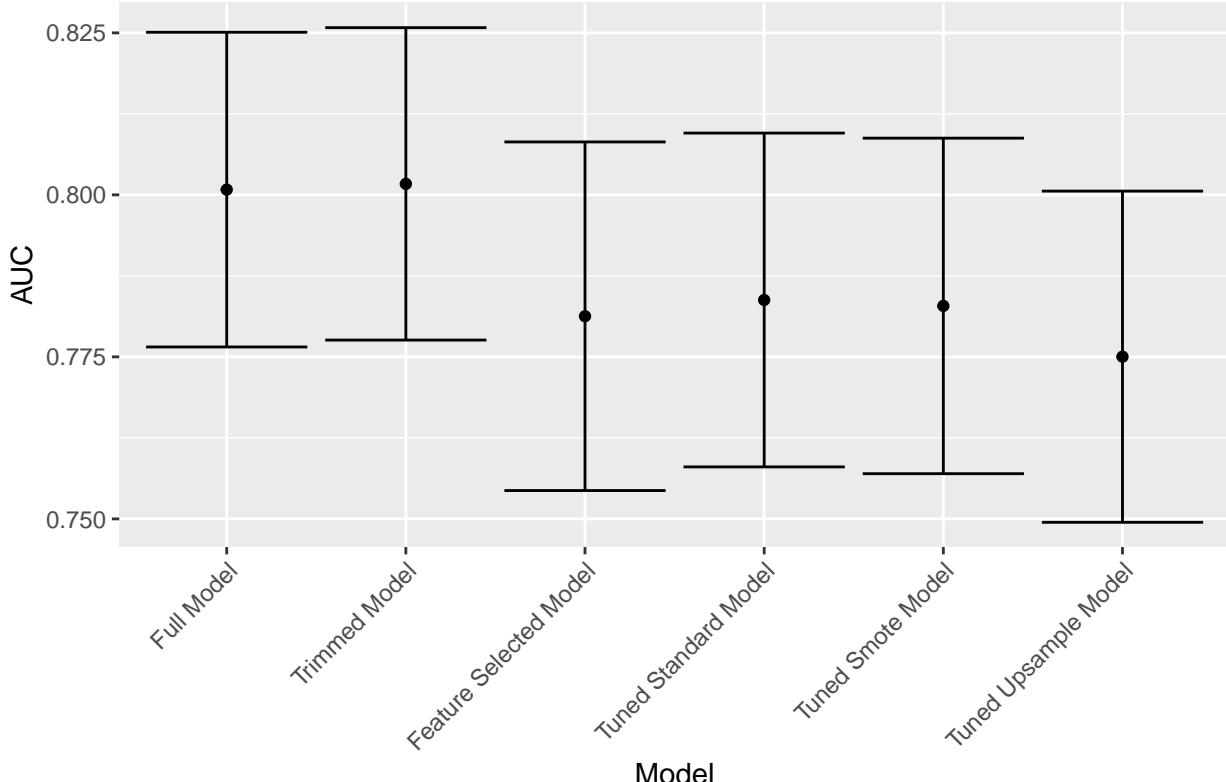
```

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$auc_upper,
  'Tuned Standard Model', as.numeric(lightgbm_auc$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))

```

death\_3year



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