Biostat 203B Homework 5

Due Mar 20 @ 11:59PM

Jiaye Tian UID: 306541095

Table of contents

Predicting ICU duration

Using the ICU cohort `mimiciv_icu_cohort.rds` you built in Homework 4, develop at least three machine learning approaches (logistic regression with enet regularization, random forest, boosting, SVM, MLP, etc) plus a model stacking approach for predicting whether a patient's ICU stay will be longer than 2 days. You should use the `los_long` variable as the outcome. You algorithms can use patient demographic information (gender, age at ICU `intime`, marital status, race), ICU admission information (first care unit), the last lab measurements before the ICU stay, and first vital measurements during ICU stay as features. You are welcome to use any feature engineering techniques you think are appropriate; but make sure to not use features that are not available at an ICU stay's `intime`. For instance, `last_careunit` cannot be used in your algorithms.

1. Data preprocessing and feature engineering.

```
sessionInfo()
```

```
R version 4.4.3 (2025-02-28)
Platform: aarch64-apple-darwin20
Running under: macOS Sonoma 14.6.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.4.3    fastmap_1.2.0     cli_3.6.4         tools_4.4.3
 [5] htmltools_0.5.8.1 rstudioapi_0.17.1 yaml_2.3.10       rmarkdown_2.29
 [9] knitr_1.49        jsonlite_1.9.1    xfun_0.51         digest_0.6.37
[13] rlang_1.1.5       evaluate_1.0.3
```

```
# Load libraries
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(rsample)
library(gtsummary)
library(GGally)
```

```
Loading required package: ggplot2
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.4     v tibble    3.2.1
v purrr     1.0.4     v tidyr     1.3.1
v readr     2.1.5
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(tidymodels)
```

```
-- Attaching packages --------------------------------------- tidymodels 1.3.0 --
v broom        1.0.7     v recipes      1.1.1
v dials        1.4.0     v tune         1.3.0
v infer        1.0.7     v workflows    1.2.0
v modeldata    1.4.0     v workflowsets 1.1.0
v parsnip      1.3.1     v yardstick    1.3.2
-- Conflicts ------------------------------------------ tidymodels_conflicts() --
x scales::discard() masks purrr::discard()
x dplyr::filter()   masks stats::filter()
x recipes::fixed()  masks stringr::fixed()
x dplyr::lag()      masks stats::lag()
x yardstick::spec() masks readr::spec()
x recipes::step()   masks stats::step()
```

```r
# Load the `mimic_icu_cohort` data.
mimic_icu_cohort <- readRDS("~/203b/hw/hw4/mimiciv_shiny/mimic_icu_cohort.rds"
                            ) |>
  select(subject_id, hadm_id, stay_id,
         gender, ageintime, marital_status, race, first_careunit,
         bicarbonate,
         chloride,
         creatinine,
         glucose,
         hematocrit,
         potassium,
         sodium,
         wbc,
         `heart rate`,
         `non invasive blood pressure systolic`,
         `non invasive blood pressure diastolic`,
         `respiratory rate`,
         `temperature fahrenheit`,
         los
  ) |>
  mutate(marital_status = as.factor(marital_status),
         race = case_when(
           str_detect(race, "ASIAN") ~ "ASIAN",
           str_detect(race, "BLACK") ~ "BLACK",
           str_detect(race, "HISPANIC") ~ "HISPANIC",
           str_detect(race, "WHITE") ~ "WHITE",
           TRUE ~ "Other" ) |>
         factor(levels = c("ASIAN", "BLACK", "HISPANIC", "WHITE", "Other")),
         first_careunit = as.factor(first_careunit),
         gender = as.factor(gender),
         los_long = as.factor(los >= 2)) |>
  select(- los) |>
  print(width = Inf)
```

```
# A tibble: 94,458 x 22
   subject_id  hadm_id  stay_id gender ageintime marital_status race
        <int>    <int>    <int> <fct>      <int> <fct>          <fct>
 1   10000032 29079034 39553978 F             52 WIDOWED        WHITE
 2   10000690 25860671 37081114 F             86 WIDOWED        WHITE
 3   10000980 26913865 39765666 F             76 MARRIED        BLACK
 4   10001217 24597018 37067082 F             55 MARRIED        WHITE
 5   10001217 27703517 34592300 F             55 MARRIED        WHITE
```

```
 6   10001725 25563031 31205490 F             46 MARRIED        WHITE
 7   10001843 26133978 39698942 M             76 SINGLE         WHITE
 8   10001884 26184834 37510196 F             77 MARRIED        BLACK
 9   10002013 23581541 39060235 F             57 SINGLE         Other
10   10002114 27793700 34672098 M             56 <NA>           Other
   first_careunit                                       bicarbonate chloride
   <fct>                                                      <dbl>    <dbl>
 1 Medical Intensive Care Unit (MICU)                            25       95
 2 Medical Intensive Care Unit (MICU)                            26      100
 3 Medical Intensive Care Unit (MICU)                            21      109
 4 Surgical Intensive Care Unit (SICU)                           22      108
 5 Surgical Intensive Care Unit (SICU)                           30      104
 6 Medical/Surgical Intensive Care Unit (MICU/SICU)              NA       98
 7 Medical/Surgical Intensive Care Unit (MICU/SICU)              28       97
 8 Medical Intensive Care Unit (MICU)                            30       88
 9 Cardiac Vascular Intensive Care Unit (CVICU)                  24      102
10 Coronary Care Unit (CCU)                                      18       NA
   creatinine glucose hematocrit potassium sodium   wbc `heart rate`
        <dbl>   <dbl>      <dbl>     <dbl>  <dbl> <dbl>        <dbl>
 1        0.7     102       41.1       6.7    126   6.9         91
 2        1        85       36.1       4.8    137   7.1         78
 3        2.3      89       27.3       3.9    144   5.3         76
 4        0.6     112       38.1       4.2    142  15.7         86
 5        0.5      87       37.4       4.1    142   5.4         79.3
 6       NA        NA       NA         4.1    139  NA           86
 7        1.3     131       31.4       3.9    138  10.4        124.
 8        1.1     141       39.7       4.5    130  12.2         49
 9        0.9     288       34.9       3.5    137   7.2         80
10        3.1      95       34.3       6.5    125  16.8        110.
   `non invasive blood pressure systolic`
                                    <dbl>
 1                                   84
 2                                  106
 3                                  154
 4                                  151
 5                                  156
 6                                   73
 7                                  110
 8                                  174.
 9                                   98.5
10                                  112
   `non invasive blood pressure diastolic` `respiratory rate`
                                     <dbl>              <dbl>
```

```
 1                                    48                 98.7
 2                                    56.5               97.7
 3                                   102                 98
 4                                    90                 98.5
 5                                    93.3               97.6
 6                                    56                 97.7
 7                                    78                 97.9
 8                                    30.5               98.1
 9                                    62                 97.2
10                                    80                 97.9
   `temperature fahrenheit` los_long
                      <dbl> <fct>
 1                     24   FALSE
 2                     24.3 TRUE
 3                     23.5 FALSE
 4                     18   FALSE
 5                     14   FALSE
 6                     19   FALSE
 7                     16.5 FALSE
 8                     13   TRUE
 9                     14   FALSE
10                     21   TRUE
# i 94,448 more rows
```

```
# Numerical summaries stratified by the outcome `los_long`.
mimic_icu_cohort |> tbl_summary(by = los_long)
```

```
14 missing rows in the "los_long" column have been removed.
```

2. Sorted stratified 50/50 split (seed 203)

2. Partition data into 50% training set and 50% test set. Stratify partitioning according to `los_long`. For grading purpose, sort the data by `subject_id`, `hadm_id`, and `stay_id` and use the seed 203 for the initial data split. Below is the sample code.

```
set.seed(203)

# sort
mimiciv_icu_cohort <- mimic_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id)
```

Characteristic	**FALSE** N = 48,107[1]	
subject_id	14,988,897 (12,506,011, 17,513,478)	15,021
hadm_id	24,954,662 (22,465,369, 27,459,051)	25,011
stay_id	35,045,664 (32,534,836, 37,518,493)	34,949
gender		
F	21,471 (45%)	
M	26,636 (55%)	
ageintime	66 (54, 77)	
marital_status		
DIVORCED	3,555 (8.0%)	
MARRIED	21,344 (48%)	
SINGLE	14,039 (31%)	
WIDOWED	5,752 (13%)	
Unknown	3,417	
race		
ASIAN	1,516 (3.2%)	
BLACK	5,452 (11%)	
HISPANIC	1,908 (4.0%)	
WHITE	32,351 (67%)	
Other	6,880 (14%)	
first_careunit		
Cardiac Vascular Intensive Care Unit (CVICU)	7,416 (15%)	
Coronary Care Unit (CCU)	5,338 (11%)	
Intensive Care Unit (ICU)	7 (<0.1%)	
Med/Surg	1 (<0.1%)	
Medical Intensive Care Unit (MICU)	10,862 (23%)	
Medical/Surgical Intensive Care Unit (MICU/SICU)	8,780 (18%)	
Medicine	1 (<0.1%)	
Medicine/Cardiology Intermediate	0 (0%)	
Neuro Intermediate	2,074 (4.3%)	
Neuro Stepdown	660 (1.4%)	
Neuro Surgical Intensive Care Unit (Neuro SICU)	795 (1.7%)	
Neurology	0 (0%)	
PACU	61 (0.1%)	
Surgery/Trauma	1 (<0.1%)	
Surgery/Vascular/Intermediate	7 (<0.1%)	
Surgical Intensive Care Unit (SICU)	6,574 (14%)	
Trauma SICU (TSICU)	5,530 (11%)	
bicarbonate	24.0 (21.0, 27.0)	
Unknown	5,277	
chloride	102 (98, 105)	
Unknown	5,167	
creatinine	1.00 (0.80, 1.40)	
Unknown	3,486	
glucose	118 (98, 154)	
Unknown	5,314	
hematocrit	36 (30, 41)	
Unknown	2,894	
potassium	4.20 (3.90, 4.60)	
Unknown	5,187	

```r
full_data <- mimic_icu_cohort |> filter(!is.na(los_long))

data_split <- initial_split(
  mimiciv_icu_cohort,
  # stratify by los_long
  strata = los_long,
  prop = 0.5
  )

data_split
```

```
<Training/Testing/Total>
<47228/47230/94458>
```

```r
train_data <- training(data_split) |>
  filter(!is.na(los_long)) |>
  select(-subject_id, -hadm_id, -stay_id)

dim(train_data)
```

```
[1] 47221    19
```

```r
prop.table(table(train_data$los_long))
```

```
   FALSE     TRUE
0.509392 0.490608
```

```r
test_data <- testing(data_split) |>
  filter(!is.na(los_long)) |>
  select(-subject_id, -hadm_id, -stay_id)

dim(test_data)
```

```
[1] 47223    19
```

```r
prop.table(table(test_data$los_long))
```

```
    FALSE      TRUE
0.5093493 0.4906507
```

```
recipe <-
  recipe(los_long ~ ., data = train_data) |>
  step_unknown(all_nominal_predictors()) |>
  step_impute_median(`heart rate`) |>
  step_impute_median(`non invasive blood pressure systolic`) |>
  step_impute_median(`non invasive blood pressure diastolic`) |>
  step_impute_median(`respiratory rate`) |>
  step_impute_median(`temperature fahrenheit`) |>
  step_mutate(across(c(bicarbonate,
                       chloride,
                       creatinine,
                       glucose,
                       potassium,
                       sodium,
                       hematocrit,
                       wbc),
                     ~ if_else(is.na(.), 1, 0),
                     .names = "{.col}_missing")) |>
  step_impute_median(c(bicarbonate,
                       chloride,
                       creatinine,
                       glucose,
                       potassium,
                       sodium,
                       hematocrit,
                       wbc)) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_numeric_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_naomit(all_outcomes()) |>
  step_naomit(all_predictors()) |>
  print()
```

```
-- Recipe ----------------------------------------------------------------------
```

```
-- Inputs

Number of variables by role

outcome:    1
predictor: 18



-- Operations

* Unknown factor level assignment for: all_nominal_predictors()

* Median imputation for: `heart rate`

* Median imputation for: `non invasive blood pressure systolic`

* Median imputation for: `non invasive blood pressure diastolic`

* Median imputation for: `respiratory rate`

* Median imputation for: `temperature fahrenheit`

* Variable mutation for: across(c(bicarbonate, chloride, creatinine, glucose,
  potassium, sodium, hematocrit, wbc), ~ if_else(is.na(.), 1, 0), .names =
  "{.col}_missing")

* Median imputation for: c(bicarbonate, chloride, creatinine, glucose,
  potassium, sodium, hematocrit, wbc)

* Dummy variables from: all_nominal_predictors()

* Zero variance filter on: all_numeric_predictors()

* Centering and scaling for: all_numeric_predictors()

* Removing rows with NA values in: all_outcomes()

* Removing rows with NA values in: all_predictors()
```

3. Train and tune the models

3. Train and tune the models using the training set.

(1) Logit Regression Model

```
# Load libraries
library(glmnet)
```

```
Loading required package: Matrix
```

```

Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```
Loaded glmnet 4.1-8
```

```
logit_mod <-
  logistic_reg(
    penalty = tune(),
    mixture = tune()) |>
  set_engine("glmnet", standardize = FALSE) |>
  print()
```

```
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
logit_wf <- workflow() |>
  add_recipe(recipe) |>
  add_model(logit_mod) |>
  print()
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: logistic_reg()

-- Preprocessor ----------------------------------------------------------------
13 Recipe Steps

* step_unknown()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_mutate()
* step_impute_median()
* step_dummy()
* step_zv()
* ...
* and 3 more steps.

-- Model -----------------------------------------------------------------------
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
param_grid_logit <- grid_regular(
  penalty(range = c(-2.8, -2.3)),
  mixture(range = c(0.2, 0.3)),
  levels = c(10, 10)
```

```
)
print(param_grid_logit)
```

```
# A tibble: 100 x 2
   penalty mixture
     <dbl>   <dbl>
 1 0.00158     0.2
 2 0.00180     0.2
 3 0.00205     0.2
 4 0.00233     0.2
 5 0.00264     0.2
 6 0.00300     0.2
 7 0.00341     0.2
 8 0.00388     0.2
 9 0.00441     0.2
10 0.00501     0.2
# i 90 more rows
```

```
# CV
set.seed(203)

folds <- vfold_cv(train_data, v = 5)
print(folds)
```

```
#  5-fold cross-validation
# A tibble: 5 x 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37777/9444]> Fold2
3 <split [37777/9444]> Fold3
4 <split [37777/9444]> Fold4
5 <split [37777/9444]> Fold5
```

```
logit_fit <- logit_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid_logit,
    metrics = metric_set(roc_auc, accuracy)
    )
print(logit_fit)
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 x 4
  splits               id    .metrics           .notes
  <list>               <chr> <list>             <list>
1 <split [37776/9445]> Fold1 <tibble [200 x 6]> <tibble [0 x 3]>
2 <split [37777/9444]> Fold2 <tibble [200 x 6]> <tibble [0 x 3]>
3 <split [37777/9444]> Fold3 <tibble [200 x 6]> <tibble [0 x 3]>
4 <split [37777/9444]> Fold4 <tibble [200 x 6]> <tibble [0 x 3]>
5 <split [37777/9444]> Fold5 <tibble [200 x 6]> <tibble [0 x 3]>
```

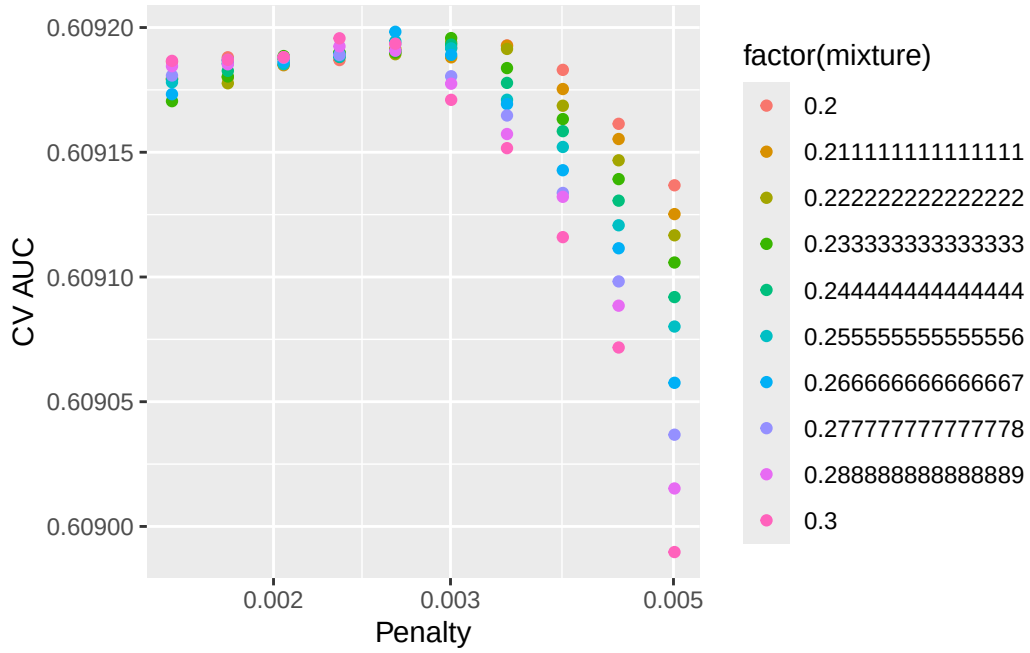
```r
system.time({
  logit_fit <- logit_wf %>%
    tune_grid(
      resamples = folds,
      grid = param_grid_logit,
      metrics = metric_set(roc_auc, accuracy)
    )
})
```

```
   user  system elapsed
 28.016   0.626  28.644
```

```r
logit_fit |>
  # aggregate metrics from K folds
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = penalty, y = mean, color = factor(mixture))) +
  geom_point() +
  labs(x = "Penalty", y = "CV AUC") +
  scale_x_log10()
```

```
# A tibble: 200 x 8
   penalty mixture .metric  .estimator  mean     n std_err
     <dbl>   <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
 1 0.00158     0.2 accuracy binary     0.582     5 0.00258
 2 0.00158     0.2 roc_auc  binary     0.609     5 0.00242
 3 0.00180     0.2 accuracy binary     0.582     5 0.00262
 4 0.00180     0.2 roc_auc  binary     0.609     5 0.00242
 5 0.00205     0.2 accuracy binary     0.582     5 0.00258
```

```
 6 0.00205     0.2 roc_auc  binary     0.609     5 0.00242
 7 0.00233     0.2 accuracy binary     0.582     5 0.00258
 8 0.00233     0.2 roc_auc  binary     0.609     5 0.00243
 9 0.00264     0.2 accuracy binary     0.582     5 0.00271
10 0.00264     0.2 roc_auc  binary     0.609     5 0.00244
   .config
   <chr>
 1 Preprocessor1_Model001
 2 Preprocessor1_Model001
 3 Preprocessor1_Model002
 4 Preprocessor1_Model002
 5 Preprocessor1_Model003
 6 Preprocessor1_Model003
 7 Preprocessor1_Model004
 8 Preprocessor1_Model004
 9 Preprocessor1_Model005
10 Preprocessor1_Model005
# i 190 more rows
```




```
logit_fit |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 x 8
```

```
  penalty mixture .metric .estimator  mean     n std_err .config
    <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 0.00264   0.267 roc_auc binary     0.609     5 0.00245 Preprocessor1_Model065
2 0.00300   0.233 roc_auc binary     0.609     5 0.00246 Preprocessor1_Model036
3 0.00233   0.3   roc_auc binary     0.609     5 0.00245 Preprocessor1_Model094
4 0.00264   0.256 roc_auc binary     0.609     5 0.00245 Preprocessor1_Model055
5 0.00300   0.222 roc_auc binary     0.609     5 0.00245 Preprocessor1_Model026
```

```r
best_logit <- logit_fit |>
  select_best(metric = "roc_auc")
print(best_logit)
```

```
# A tibble: 1 x 3
  penalty mixture .config
    <dbl>   <dbl> <chr>
1 0.00264   0.267 Preprocessor1_Model065
```

```r
# Final workflow
final_wf_logit <- logit_wf |>
  finalize_workflow(best_logit)
print(final_wf_logit)
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: logistic_reg()

-- Preprocessor ----------------------------------------------------------------
13 Recipe Steps

* step_unknown()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_mutate()
* step_impute_median()
* step_dummy()
* step_zv()
* ...
* and 3 more steps.
```

```
-- Model ---------------------------------------------------------------------
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = 0.0026437611857491
  mixture = 0.266666666666667

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

(2) Boosting Model

```
# Load libraries
library(xgboost)
```

```
Attaching package: 'xgboost'


The following object is masked from 'package:dplyr':

    slice
```

```
gb_mod <-
  boost_tree(
    mode = "classification",
    trees = 1000,
    tree_depth = tune(),
    learn_rate = tune()
  ) |>
  set_engine("xgboost", early_stopping_rounds = 50)
print(gb_mod)
```

```
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 1000
  tree_depth = tune()
```

```
  learn_rate = tune()

Engine-Specific Arguments:
  early_stopping_rounds = 50

Computational engine: xgboost
```

```
gb_wf <- workflow() |>
  add_recipe(recipe) |>
  add_model(gb_mod)
print(gb_wf)
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: boost_tree()

-- Preprocessor ----------------------------------------------------------------
13 Recipe Steps

* step_unknown()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_mutate()
* step_impute_median()
* step_dummy()
* step_zv()
* ...
* and 3 more steps.

-- Model -----------------------------------------------------------------------
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 1000
  tree_depth = tune()
  learn_rate = tune()

Engine-Specific Arguments:
  early_stopping_rounds = 50
```

```
Computational engine: xgboost
```

```r
param_grid_gb <- grid_regular(
  tree_depth(range = c(2L, 4L)),
  learn_rate(range = c(-1.3, -1.0), trans = log10_trans()),
  levels = c(3, 10)
  )
print(param_grid_gb)
```

```
# A tibble: 30 x 2
   tree_depth learn_rate
        <int>      <dbl>
 1          2     0.0501
 2          3     0.0501
 3          4     0.0501
 4          2     0.0541
 5          3     0.0541
 6          4     0.0541
 7          2     0.0584
 8          3     0.0584
 9          4     0.0584
10          2     0.0631
# i 20 more rows
```

```r
set.seed(203)
folds <- vfold_cv(train_data, v = 5)
folds
```

```
#  5-fold cross-validation
# A tibble: 5 x 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37777/9444]> Fold2
3 <split [37777/9444]> Fold3
4 <split [37777/9444]> Fold4
5 <split [37777/9444]> Fold5
```

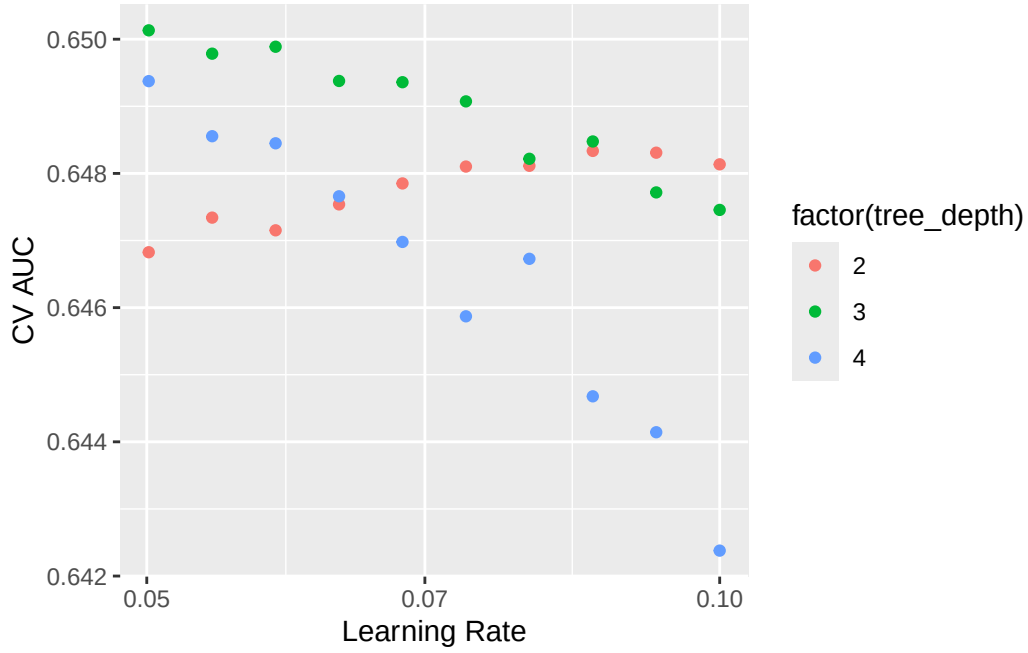
```r
gb_fit <- gb_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid_gb,
    metrics = metric_set(roc_auc, accuracy)
    )
print(gb_fit)
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 x 4
  splits               id    .metrics          .notes
  <list>               <chr> <list>            <list>
1 <split [37776/9445]> Fold1 <tibble [60 x 6]> <tibble [0 x 3]>
2 <split [37777/9444]> Fold2 <tibble [60 x 6]> <tibble [0 x 3]>
3 <split [37777/9444]> Fold3 <tibble [60 x 6]> <tibble [0 x 3]>
4 <split [37777/9444]> Fold4 <tibble [60 x 6]> <tibble [0 x 3]>
5 <split [37777/9444]> Fold5 <tibble [60 x 6]> <tibble [0 x 3]>
```

```r
gb_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = learn_rate, y = mean, color = factor(tree_depth))) +
  geom_point() +
  labs(x = "Learning Rate", y = "CV AUC") +
  scale_x_log10()
```

```
# A tibble: 60 x 8
   tree_depth learn_rate .metric  .estimator  mean     n std_err
        <int>      <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
 1          2     0.0501 accuracy binary     0.607     5 0.00163
 2          2     0.0501 roc_auc  binary     0.647     5 0.00213
 3          3     0.0501 accuracy binary     0.609     5 0.00195
 4          3     0.0501 roc_auc  binary     0.650     5 0.00224
 5          4     0.0501 accuracy binary     0.605     5 0.00284
 6          4     0.0501 roc_auc  binary     0.649     5 0.00241
 7          2     0.0541 accuracy binary     0.608     5 0.00191
 8          2     0.0541 roc_auc  binary     0.647     5 0.00217
 9          3     0.0541 accuracy binary     0.608     5 0.00211
10          3     0.0541 roc_auc  binary     0.650     5 0.00245
```

```
   .config
   <chr>
 1 Preprocessor1_Model01
 2 Preprocessor1_Model01
 3 Preprocessor1_Model02
 4 Preprocessor1_Model02
 5 Preprocessor1_Model03
 6 Preprocessor1_Model03
 7 Preprocessor1_Model04
 8 Preprocessor1_Model04
 9 Preprocessor1_Model05
10 Preprocessor1_Model05
# i 50 more rows
```




```
gb_fit |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 x 8
  tree_depth learn_rate .metric .estimator  mean     n std_err .config
       <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1          3     0.0501 roc_auc binary     0.650     5 0.00224 Preprocessor1_Mo~
2          3     0.0584 roc_auc binary     0.650     5 0.00242 Preprocessor1_Mo~
```

```
3          3     0.0541 roc_auc binary     0.650     5 0.00245 Preprocessor1_Mo~
4          3     0.0631 roc_auc binary     0.649     5 0.00246 Preprocessor1_Mo~
5          4     0.0501 roc_auc binary     0.649     5 0.00241 Preprocessor1_Mo~
```

```r
best_gb <- gb_fit |>
  select_best(metric = "roc_auc")
print(best_gb)
```

```
# A tibble: 1 x 3
  tree_depth learn_rate .config
       <int>      <dbl> <chr>
1          3     0.0501 Preprocessor1_Model02
```

```r
# Final workflow
final_wf_gb <- gb_wf |>
  finalize_workflow(best_gb)
print(final_wf_gb)
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: boost_tree()

-- Preprocessor ----------------------------------------------------------------
13 Recipe Steps

* step_unknown()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_mutate()
* step_impute_median()
* step_dummy()
* step_zv()
* ...
* and 3 more steps.

-- Model -----------------------------------------------------------------------
Boosted Tree Model Specification (classification)
```

```
Main Arguments:
  trees = 1000
  tree_depth = 3
  learn_rate = 0.0501187233627272

Engine-Specific Arguments:
  early_stopping_rounds = 50

Computational engine: xgboost
```

(3) Random Forest Model

```
# Load libraries
library(ranger)

rf_mod <-
  rand_forest(
    mode = "classification",
    # Number of predictors randomly sampled in each split
    mtry = tune(),
    # Number of trees in ensemble
    trees = tune()
  ) |>
  set_engine("ranger", importance = "impurity")
print(rf_mod)
```

```
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```
rf_wf <- workflow() |>
  add_recipe(recipe) |>
  add_model(rf_mod)
print(rf_wf)
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
13 Recipe Steps

* step_unknown()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_mutate()
* step_impute_median()
* step_dummy()
* step_zv()
* ...
* and 3 more steps.

-- Model -----------------------------------------------------------------------
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```r
param_grid_rf <- grid_regular(
  trees(range = c(300L, 400L)),
  mtry(range = c(7L, 11L)),
  levels = c(5, 5)
  )
print(param_grid_rf)
```

```
# A tibble: 25 x 2
   trees  mtry
   <int> <int>
```

```
 1   300     7
 2   325     7
 3   350     7
 4   375     7
 5   400     7
 6   300     8
 7   325     8
 8   350     8
 9   375     8
10   400     8
# i 15 more rows
```

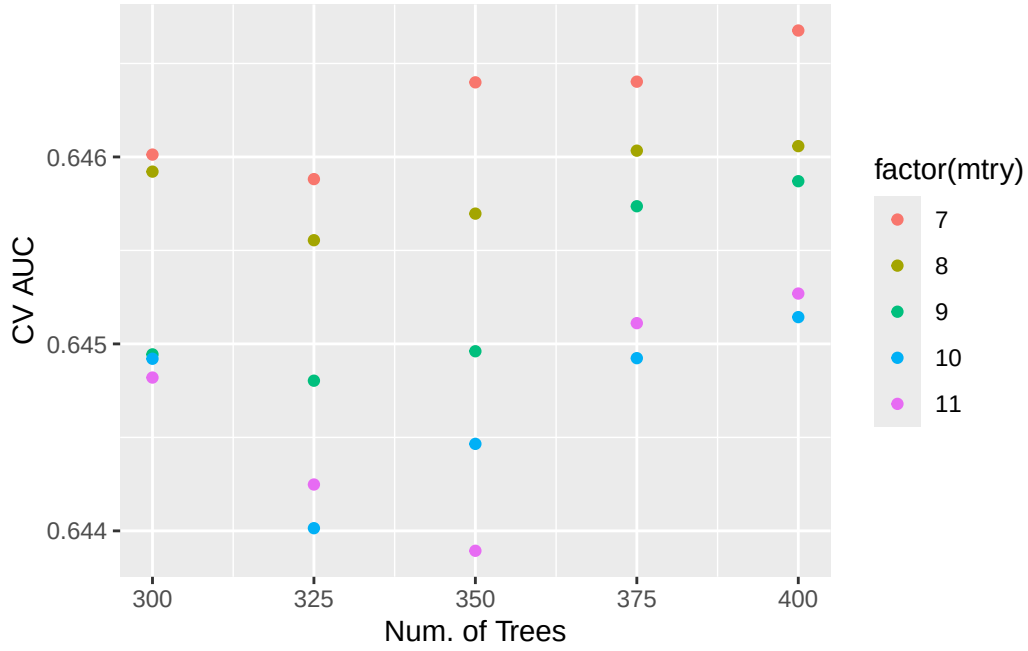
```
set.seed(203)
folds <- vfold_cv(train_data, v = 5)
folds
```

```
#  5-fold cross-validation
# A tibble: 5 x 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37777/9444]> Fold2
3 <split [37777/9444]> Fold3
4 <split [37777/9444]> Fold4
5 <split [37777/9444]> Fold5
```

```
library(future)
plan(multisession, workers = parallel::detectCores() - 1)
rf_fit <- rf_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid_rf,
    metrics = metric_set(roc_auc, accuracy)
    )

rf_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = trees, y = mean, color = factor(mtry))) +
  geom_point() +
```

```
  # geom_line() +
  labs(x = "Num. of Trees", y = "CV AUC")
```

```
# A tibble: 50 x 8
    mtry trees .metric  .estimator  mean     n  std_err .config
   <int> <int> <chr>    <chr>      <dbl> <int>    <dbl> <chr>
 1     7   300 accuracy binary     0.606     5 0.000735 Preprocessor1_Model01
 2     7   300 roc_auc  binary     0.646     5 0.00170  Preprocessor1_Model01
 3     7   325 accuracy binary     0.604     5 0.00133  Preprocessor1_Model02
 4     7   325 roc_auc  binary     0.646     5 0.00193  Preprocessor1_Model02
 5     7   350 accuracy binary     0.603     5 0.00140  Preprocessor1_Model03
 6     7   350 roc_auc  binary     0.646     5 0.00203  Preprocessor1_Model03
 7     7   375 accuracy binary     0.604     5 0.00145  Preprocessor1_Model04
 8     7   375 roc_auc  binary     0.646     5 0.00141  Preprocessor1_Model04
 9     7   400 accuracy binary     0.604     5 0.00115  Preprocessor1_Model05
10     7   400 roc_auc  binary     0.647     5 0.00148  Preprocessor1_Model05
# i 40 more rows
```




```
rf_fit |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 x 8
   mtry trees .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     7   400 roc_auc binary     0.647     5 0.00148 Preprocessor1_Model05
2     7   375 roc_auc binary     0.646     5 0.00141 Preprocessor1_Model04
3     7   350 roc_auc binary     0.646     5 0.00203 Preprocessor1_Model03
4     8   400 roc_auc binary     0.646     5 0.00138 Preprocessor1_Model10
5     8   375 roc_auc binary     0.646     5 0.00171 Preprocessor1_Model09
```

```
best_rf <- rf_fit |>
  select_best(metric = "roc_auc")
print(best_rf)
```

```
# A tibble: 1 x 3
   mtry trees .config
  <int> <int> <chr>
1     7   400 Preprocessor1_Model05
```

```
# Final workflow
final_wf_rf <- rf_wf |>
  finalize_workflow(best_rf)
print(final_wf_rf)
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
13 Recipe Steps

* step_unknown()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_impute_median()
* step_mutate()
* step_impute_median()
* step_dummy()
* step_zv()
* ...
```

```
* and 3 more steps.

-- Model ---------------------------------------------------------------------
Random Forest Model Specification (classification)

Main Arguments:
  mtry = 7
  trees = 400

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

4. Test evaluation, performance comparison, and feature importance.

4. Compare model classification performance on the test set. Report both the area under ROC curve and accuracy for each machine learning algorithm and the model stacking. Interpret the results. What are the most important features in predicting long ICU stays? How do the models compare in terms of performance and interpretability?

4.1 Model Performance Comparison

We evaluated the performance of three different models: Logistic Regression, Boosting (XGBoost), and Random Forest—against the test data. The outcome is that: Logistic Regression: Accuracy 0.584, AUC 0.614 Boosting (XGBoost): Accuracy 0.609, AUC 0.654 Random Forest: Accuracy 0.606, AUC 0.647 In ROC AUC, the Boosting model did slightly better to distinguish patients with long ICU length of stay (los_long = TRUE) with an AUC measure of approximately 0.654. The Random Forest model followed closely with an AUC measure of approximately 0.647, whereas the Logistic Regression model had a comparatively lower AUC of 0.614. In Accuracy, all three models had measures ranging from approximately 0.58 to 0.61, indicating very minimal difference in general accuracy of classification.

```
# Logistic
# Fit the whole training set, then predict the test cases
final_fit_logit <-
  final_wf_logit |>
  last_fit(data_split)
print(final_fit_logit)
```

```
# Resampling results
```

```
# Manual resampling
# A tibble: 1 x 6
  splits                id             .metrics .notes   .predictions .workflow
  <list>                <chr>          <list>   <list>   <list>       <list>
1 <split [47228/47230]> train/test sp~ <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
logit_metrics <- final_fit_logit |>
  collect_metrics()
print(logit_metrics)
```

```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.584 Preprocessor1_Model1
2 roc_auc     binary         0.614 Preprocessor1_Model1
3 brier_class binary         0.240 Preprocessor1_Model1
```

```
# Boosting
# Fit the whole training set, then predict the test cases
final_fit_gb <-
  final_wf_gb |>
  last_fit(data_split)
final_fit_gb
```

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits                id             .metrics .notes   .predictions .workflow
  <list>                <chr>          <list>   <list>   <list>       <list>
1 <split [47228/47230]> train/test sp~ <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
gb_metrics <- final_fit_gb |>
  collect_metrics()
print(gb_metrics)
```

```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
```

```
1 accuracy    binary         0.609 Preprocessor1_Model1
2 roc_auc     binary         0.654 Preprocessor1_Model1
3 brier_class binary         0.232 Preprocessor1_Model1
```

```
# Random Forest
# Fit the whole training set, then predict the test cases
final_fit_rf <-
  final_wf_rf |>
  last_fit(data_split)
final_fit_rf
```

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits                id             .metrics .notes   .predictions .workflow
  <list>                <chr>          <list>   <list>   <list>       <list>
1 <split [47228/47230]> train/test sp~ <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
rf_metrics <- final_fit_rf |>
  collect_metrics()
print(rf_metrics)
```

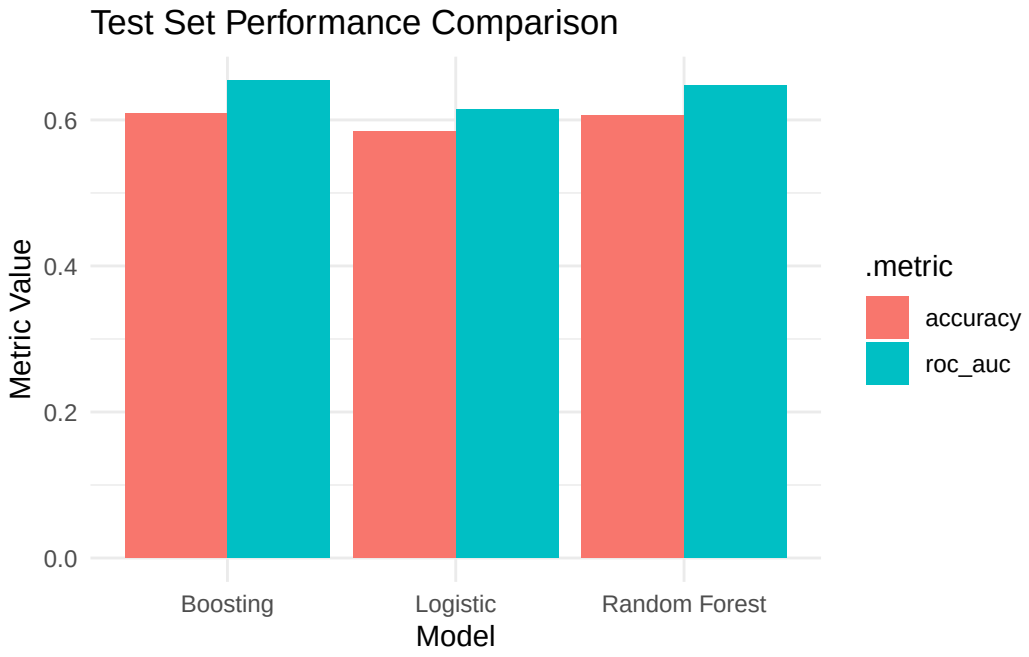
```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.606 Preprocessor1_Model1
2 roc_auc     binary         0.647 Preprocessor1_Model1
3 brier_class binary         0.234 Preprocessor1_Model1
```

```
model_perf <- bind_rows(
  rf_metrics %>% mutate(model = "Random Forest"),
  gb_metrics %>% mutate(model = "Boosting"),
  logit_metrics %>% mutate(model = "Logistic")
) %>% filter(.metric %in% c("roc_auc", "accuracy"))
print(model_perf)
```

```
# A tibble: 6 x 5
  .metric  .estimator .estimate .config              model
  <chr>    <chr>          <dbl> <chr>                <chr>
```

```
1 accuracy binary         0.606 Preprocessor1_Model1 Random Forest
2 roc_auc  binary         0.647 Preprocessor1_Model1 Random Forest
3 accuracy binary         0.609 Preprocessor1_Model1 Boosting
4 roc_auc  binary         0.654 Preprocessor1_Model1 Boosting
5 accuracy binary         0.584 Preprocessor1_Model1 Logistic
6 roc_auc  binary         0.614 Preprocessor1_Model1 Logistic
```

```r
model_perf <- model_perf %>% rename(metric_mean = .estimate)

ggplot(model_perf, aes(x = model, y = metric_mean, fill = .metric)) +
  geom_col(position = "dodge") +
  labs(title = "Test Set Performance Comparison", x = "Model", y = "Metric Value") +
  theme_minimal()
```



4.2 Model Interpretability and Feature Importance

Logistic Regression: From the coefficients, we can get a rough intuition for the direction and magnitude of the effect of each feature on elevated ICU stay. Still, with its linear assumption, it could be inadequate in describing sophisticated nonlinear interactions or relationships between features.

Boosting / Random Forest: According to the VIP (Variable Importance Plot), the features "non-invasive blood pressure systolic," "heart rate," "respiratory rate," and "hematocrit" are

top-ranked for their importance. This is most closely congruent with the traits showing highest absolute coefficients in the Logistic model, thus further validating the important role of these variables in predicting extended ICU stays. While tree-based models tend to have superior predictive performance, they are typically less interpretable. This necessitates the application of variable importance plots, partial dependence plots, or other tools to interpret the decision-making.

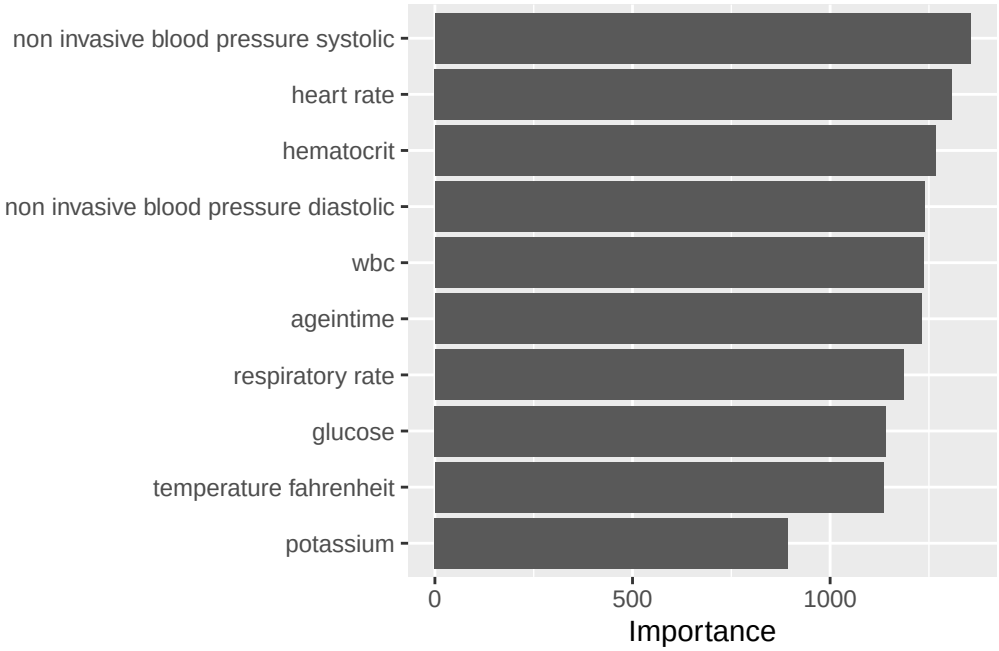
```
library(broom)
logit_object <- extract_fit_parsnip(final_fit_logit$.workflow[[1]])$fit
tidy(logit_object) %>%
  arrange(desc(abs(estimate))) %>%
  head(10)
```

```
# A tibble: 10 x 5
   term                                step estimate  lambda dev.ratio
   <chr>                              <dbl>    <dbl>   <dbl>     <dbl>
 1 first_careunit_Neuro.Intermediate     38    0.183 0.00488    0.0276
 2 first_careunit_Neuro.Intermediate     39    0.183 0.00445    0.0277
 3 first_careunit_Neuro.Intermediate     40    0.183 0.00405    0.0277
 4 first_careunit_Neuro.Intermediate     41    0.183 0.00369    0.0278
 5 first_careunit_Neuro.Intermediate     37    0.183 0.00536    0.0275
 6 first_careunit_Neuro.Intermediate     36    0.183 0.00588    0.0274
 7 first_careunit_Neuro.Intermediate     42    0.183 0.00337    0.0278
 8 first_careunit_Neuro.Intermediate     35    0.183 0.00645    0.0273
 9 first_careunit_Neuro.Intermediate     43    0.183 0.00307    0.0279
10 first_careunit_Neuro.Intermediate     50    0.183 0.00160    0.0280
```

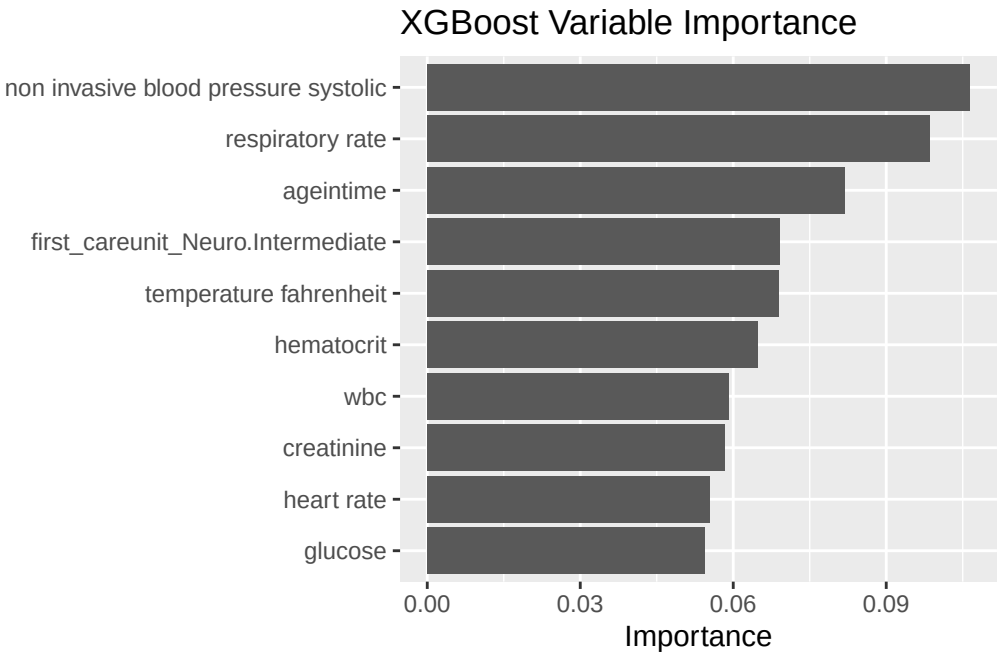
```
library(vip)
```

```
Attaching package: 'vip'

The following object is masked from 'package:utils':

    vi
```

```
rf_object <- extract_fit_parsnip(final_fit_rf$.workflow[[1]])$fit
vip(rf_object, num_features = 10, geom = "col", main = "Random Forest Importance")
```




```
gb_object <- extract_fit_parsnip(final_fit_gb$.workflow[[1]])$fit
p <- vip(gb_object, num_features = 10, geom = "col")
p + ggtitle("XGBoost Variable Importance")
```



4.3 Summary

From the test set results, the Boosting model delivered the best performance on the test set, with an AUC of (0.654); Random Forest (AUC=0.647) was very close behind, and Logistic Regression (AUC=0.614) lagged somewhat. However, it should be noted that the Accuracy of all three models was very similar, within roughly (0.58–0.61), suggesting they all classify almost equally good. In the clinical context, if higher interpretability was desired, then Logistic Regression is the simplest approach; otherwise, Boosting or Random Forest might be the models of interest for higher predictive performance. The clinically interesting features such as "non invasive blood pressure systolic" and the "heart rate" fit well with overall clinical intuition, and therefore, these two features are likely to be significant contributors to a patient's probability of a longer ICU stay.