Biostat 203B Homework 5

Due Mar 20 @ 11:59PM

Wenqiang Ge UID:106371961

Table of contents

0.1 Predicting ICU duration 1

0.1 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.

```
# Load libraries
library(GGally)
```

```
Loading required package: ggplot2
```

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

```
library(gtsummary)
library(ranger)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
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
-- 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 beco
```

```
library(tidymodels)
```

```
-- Attaching packages ---------------------------------------- tidymodels 1.3.0 --
v broom        1.0.7     v rsample      1.2.1
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      1.3.0     v yardstick    1.3.2
v recipes      1.1.1
-- 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()
```

```
library(xgboost)
```

```

Attaching package: 'xgboost'

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

    slice
```

```
library(doParallel)
```

```
Loading required package: foreach

Attaching package: 'foreach'

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

    accumulate, when

Loading required package: iterators
Loading required package: parallel
```

```
library(stacks)
library(keras)
```

```
Attaching package: 'keras'

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

    get_weights
```

```
library(dplyr)
library(e1071)
```

```
Attaching package: 'e1071'

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

    tune

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

    permutations

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

    tune
```

```
library(rsample)
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
```

```
library(vip)
```

```
Attaching package: 'vip'

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

    vi
```

```
library(ggplot2)
library(gt)
library(lubridate)
```

```
rmarkdown::render("hw5.qmd", envir = globalenv())
```

```
processing file: hw5.qmd
```

```
  |
  |                                                      |   0%
  |
  |.                                                     |   1%
  |
  |.                                                     |   2% [unnamed-chunk-22]
  |
  |..                                                    |   3%
  |
  |..                                                    |   4% [unnamed-chunk-23]
  |
  |...                                                   |   5%
  |
  |...                                                   |   6% [unnamed-chunk-24]
  |
  |....                                                  |   8%
  |
  |....                                                  |   9% [unnamed-chunk-25]
  |
  |.....                                                 |  10%
  |
  |.....                                                 |  11% [unnamed-chunk-26]
  |
  |......                                                |  12%
  |
  |.......                                               |  13% [unnamed-chunk-27]
  |
  |.......                                               |  14%
  |
  |........                                              |  15% [unnamed-chunk-28]
  |
  |........                                              |  16%
  |
  |.........                                             |  17% [unnamed-chunk-29]
  |
```

```
|.........                                |  18%
|
|..........                               |  19% [unnamed-chunk-30]
|
|..........                               |  20%
|
|...........                              |  22% [unnamed-chunk-31]
|
|............                             |  23%
|
|............                             |  24% [unnamed-chunk-32]
|
|.............                            |  25%
|
|.............                            |  26% [unnamed-chunk-33]
|
|..............                           |  27%
|
|..............                           |  28% [unnamed-chunk-34]

|
|...............                          |  29%
|
|...............                          |  30% [unnamed-chunk-35]
|
|................                         |  31%
|
|................                         |  32% [unnamed-chunk-36]
|
|.................                        |  33%
|
|.................                        |  34% [unnamed-chunk-37]
|
|..................                       |  35%
|
|..................                       |  37% [unnamed-chunk-38]
|
|...................                      |  38%
|
|...................                      |  39% [unnamed-chunk-39]

|
|...................                      |  40%
|
|....................                     |  41% [unnamed-chunk-40]
```

```
|
|.....                                 |  42%
|
|.....                                 |  43% [unnamed-chunk-41]
|
|.....                                 |  44%
|
|.....                                 |  45% [unnamed-chunk-42]
|
|.....                                 |  46%
|
|.....                                 |  47% [unnamed-chunk-43]
|
|.....                                 |  48%
|
|.....                                 |  49% [unnamed-chunk-44]
|
|.....                                 |  51%
|
|.....                                 |  52% [unnamed-chunk-45]
|
|.....                                 |  53%
|
|.....                                 |  54% [unnamed-chunk-46]
|
|.....                                 |  55%
|
|.....                                 |  56% [unnamed-chunk-47]

|
|.....                                 |  57%
|
|.....                                 |  58% [unnamed-chunk-48]
|
|.....                                 |  59%
|
|.....                                 |  60% [unnamed-chunk-49]
|
|.....                                 |  61%
|
|.....                                 |  62% [unnamed-chunk-50]
|
|.....                                 |  63%
|
|.....                                 |  65% [unnamed-chunk-51]
```

```
|
|.....................................  |  66%
|
|.....................................  |  67% [unnamed-chunk-52]
|
|.....................................  |  68%
|
|.....................................  |  69% [unnamed-chunk-53]

|
|.....................................  |  70%
|
|.....................................  |  71% [unnamed-chunk-54]
|
|.....................................  |  72%
|
|.....................................  |  73% [unnamed-chunk-55]
|
|.....................................  |  74%
|
|.....................................  |  75% [unnamed-chunk-56]
|
|.....................................  |  76%
|
|.....................................  |  77% [unnamed-chunk-57]
|
|.....................................  |  78%
|
|.....................................  |  80% [unnamed-chunk-58]
|
|.....................................  |  81%
|
|.....................................  |  82% [unnamed-chunk-59]
|
|.....................................  |  83%
|
|.....................................  |  84% [unnamed-chunk-60]

|
|.....................................  |  85%
|
|.....................................  |  86% [unnamed-chunk-61]
|
|.....................................  |  87%
|
```

```
  |.........................................      |  88% [unnamed-chunk-62]
  |
  |.........................................      |  89%
  |
  |.........................................      |  90% [unnamed-chunk-63]
  |
  |.........................................      |  91%
  |
  |.........................................      |  92% [unnamed-chunk-64]
  |
  |.........................................      |  94%
  |
  |.........................................      |  95% [unnamed-chunk-65]
  |
  |.........................................      |  96%
  |
  |.........................................      |  97% [unnamed-chunk-66]


  |
  |.........................................      |  98%
  |
  |.........................................      |  99% [unnamed-chunk-67]
  |
  |.........................................      | 100%
```

```
output file: hw5.knit.md
```

```
/usr/lib/rstudio-server/bin/quarto/bin/tools/x86_64/pandoc +RTS -K512m -RTS hw5.knit.md --to
```

```
Output created: hw5.html
```

```r
#loading data
mimic_icu_cohort <- readRDS("../hw4/mimiciv_shiny/mimic_icu_cohort.rds")
lab <- c("creatinine", "potassium", "sodium", "chloride", "bicarbonate",
              "hematocrit", "glucose", "wbc")

vital <- c("heart_rate", "non-invasive_blood_pressure_systolic",
                "non-invasive_blood_pressure_diastolic",
                "temperature_fahrenheit", "respiratory_rate")

#Select the variables
mimic_icu_cohort <- mimic_icu_cohort |>
  select(c("subject_id", "hadm_id", "stay_id",
           "gender","first_careunit",
            "insurance","marital_status",
```

```
           "race","age_intime",
            all_of(lab), all_of(vital),
           "los_long")) |>
  arrange("subject_id", "hadm_id", "stay_id")
```

```
#Convert gender to factor variables
mimic_icu_cohort$gender <- ifelse(mimic_icu_cohort$gender == "M",0,1)
mimic_icu_cohort$gender <- factor(mimic_icu_cohort$gender)

# For character or factor variables, replace NA with 0, and label other unique values
mimic_icu_cohort[] <- lapply(mimic_icu_cohort, function(x) {
  # Only modify character or factor columns, but skip 'gender'
  if (is.factor(x) || is.character(x)) {
    # Skip the 'gender' column
    if (!"gender" %in% names(mimic_icu_cohort)[
      which(sapply(mimic_icu_cohort, identical, x))
      ]) {
      # Convert to character type if it is not gender
      x <- as.character(x)

      # Replace NA with "0"
      x[is.na(x)] <- "0"

      # Convert back to factor and then numeric
      x <- factor(x)
      return(as.numeric(x))  # Convert factor levels to numeric
    }
  }
  return(x)  # Leave other columns unchanged
})
```

```
# Check for outliers in numeric variables
mimic_icu_cohort[] <- lapply(mimic_icu_cohort, function(x) {
  if (is.numeric(x)) {
    # Calculate the first (Q1) and third (Q3) quartiles, and the IQR (Interquartile Range)
    Q1 <- quantile(x, 0.25, na.rm = TRUE)
    Q3 <- quantile(x, 0.75, na.rm = TRUE)
    IQR_value <- Q3 - Q1

    # Define the lower and upper limits for outliers using the IQR method
    lower_limit <- Q1 - 1.5 * IQR_value
    upper_limit <- Q3 + 1.5 * IQR_value

    # Replace values outside the defined limits with NA (outliers)
    x[x < lower_limit | x > upper_limit] <- NA
  }
```

```
  return(x)
})
```

```
#Replace NA values with the mean of the column
mimic_icu_cohort[] <- lapply(mimic_icu_cohort, function(x) {
  if (is.numeric(x)) {
    x[is.na(x)] <- mean(x, na.rm = TRUE)  # Replace NA with the mean value
  }
  return(x)
})
```

```
# Convert los_long to a factor
mimic_icu_cohort$los_long <- factor(mimic_icu_cohort$los_long, levels = c(FALSE, TRUE))

# Remove rows where los_long is NA
mimic_icu_cohort <- mimic_icu_cohort %>%
  drop_na(los_long)

#Convert other variables to factors
mimic_icu_cohort$marital_status <- factor(mimic_icu_cohort$marital_status)
mimic_icu_cohort$race <- factor(mimic_icu_cohort$race)
mimic_icu_cohort$first_careunit <- factor(mimic_icu_cohort$first_careunit)
mimic_icu_cohort$insurance <- factor(mimic_icu_cohort$insurance)

head(mimic_icu_cohort)
```

```
# A tibble: 6 x 23
  subject_id  hadm_id  stay_id gender first_careunit insurance marital_status
       <dbl>    <dbl>    <dbl> <fct>  <fct>          <fct>     <fct>
1   10000032 29079034 39553978 1      2              2         5
2   10000690 25860671 37081114 1      2              3         5
3   10000980 26913865 39765666 1      2              3         3
4   10001217 24597018 37067082 1      5              6         3
5   10001217 27703517 34592300 1      5              6         3
6   10001725 25563031 31205490 1      3              6         3
# i 16 more variables: race <fct>, age_intime <dbl>, creatinine <dbl>,
#   potassium <dbl>, sodium <dbl>, chloride <dbl>, bicarbonate <dbl>,
#   hematocrit <dbl>, glucose <dbl>, wbc <dbl>, heart_rate <dbl>,
#   `non-invasive_blood_pressure_systolic` <dbl>,
#   `non-invasive_blood_pressure_diastolic` <dbl>,
#   temperature_fahrenheit <dbl>, respiratory_rate <dbl>, los_long <fct>
```

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.

0.1.0.1 Initial split into test and non-test sets

```
set.seed(203)

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

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

mimic_train <- training(data_split) |>
  select(-subject_id, -hadm_id, -stay_id)
dim(mimic_train )
```

```
[1] 47221    20
```

```
mimic_test <- testing(data_split) |>
  select(-subject_id, -hadm_id, -stay_id)
dim(mimic_test)
```

```
[1] 47223    20
```

0.1.0.2 Recipe

```
data_recipe <-
  recipe(
    los_long ~ .,
    data = mimic_train
    ) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_numeric_predictors()) |>
  step_normalize(all_numeric_predictors())
data_recipe
```

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

-- Inputs

Number of variables by role

outcome:    1
```

```
predictor: 19


-- Operations

* Dummy variables from: all_nominal_predictors()

* Zero variance filter on: all_numeric_predictors()

* Centering and scaling for: all_numeric_predictors()
```

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?

0.1.1 Stacking model

0.1.1.1 Model & Workflow

```
logit_res <- read_rds("logit_res.rds")
bt_res <- read_rds("bt_fit.rds")
rf_res <- read_rds("rf_res.rds")
```

```
library(doParallel)
registerDoParallel(cores = 4)
```

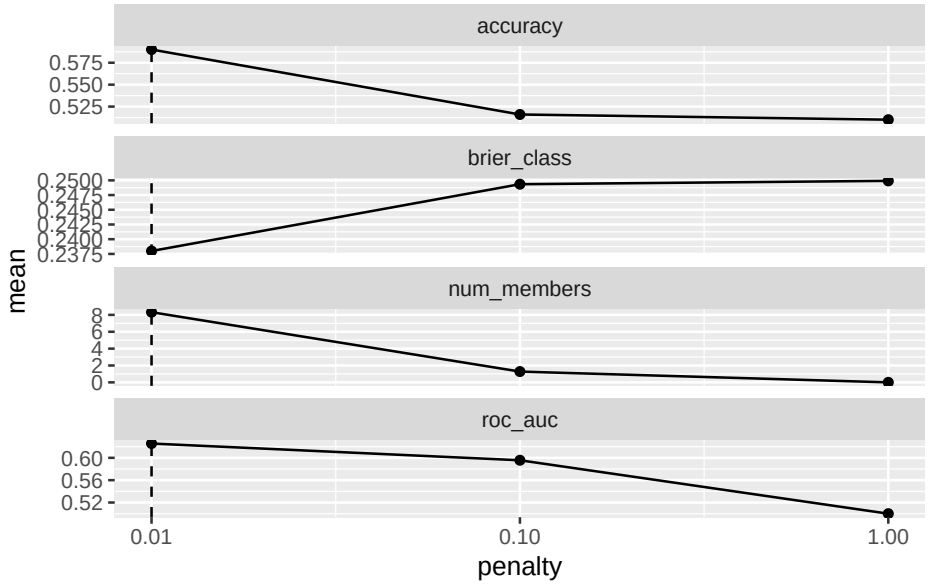
```
# build the stacked ensemble
if (file.exists("model_stack.rds")) {
  model_stack_res <- read_rds("model_stack.rds")
} else {
  model_stack_res <-
    # initialize the stack
    stacks() |>
    # add candidate members
    add_candidates(logit_res) |>
    add_candidates(bt_res) |>
    add_candidates(rf_res) |>
    # determine how to combine their predictions
    blend_predictions(
      penalty = 10^(-2:0),
      metrics = c("roc_auc")
      ) |>
    # fit the candidates with nonzero stacking coefficients
    fit_members()
  write_rds(model_stack_res, "model_stack.rds")
}
model_stack_res
```

```
-- A stacked ensemble model -------------------------------------


Out of 21 possible candidate members, the ensemble retained 9.

Penalty: 0.01.

Mixture: 1.


The 9 highest weighted member classes are:
```

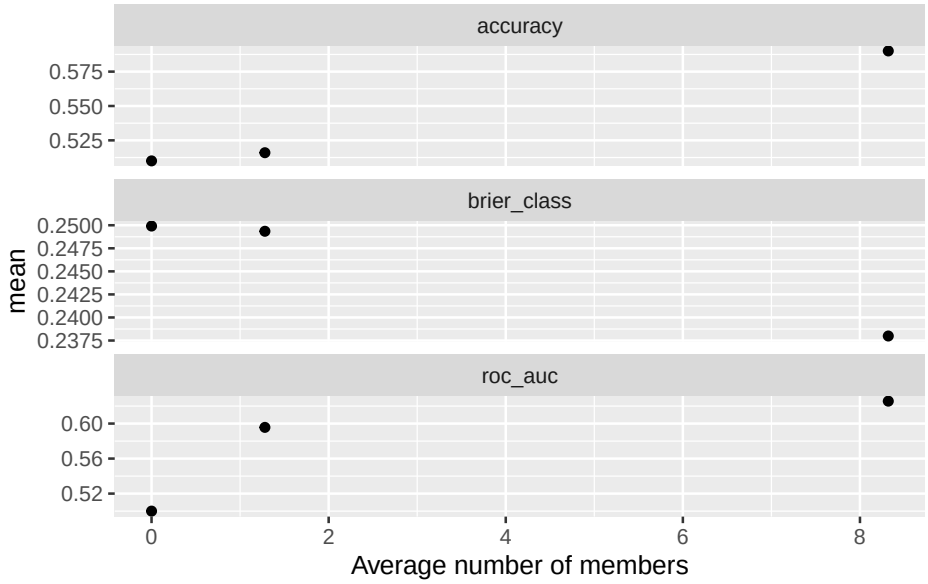
```
# A tibble: 9 x 3
  member                   type         weight
  <chr>                    <chr>         <dbl>
1 .pred_TRUE_rf_res_1_2    rand_forest  1.41
2 .pred_TRUE_rf_res_1_4    rand_forest  0.963
3 .pred_TRUE_bt_res_1_7    boost_tree   0.810
4 .pred_TRUE_rf_res_1_3    rand_forest  0.416
5 .pred_TRUE_logit_res_1_1 logistic_reg 0.329
6 .pred_TRUE_bt_res_1_6    boost_tree   0.244
7 .pred_TRUE_bt_res_1_8    boost_tree   0.225
8 .pred_TRUE_rf_res_1_1    rand_forest  0.176
9 .pred_TRUE_bt_res_1_9    boost_tree   0.0943
```

0.1.1.2 Visualize the final model

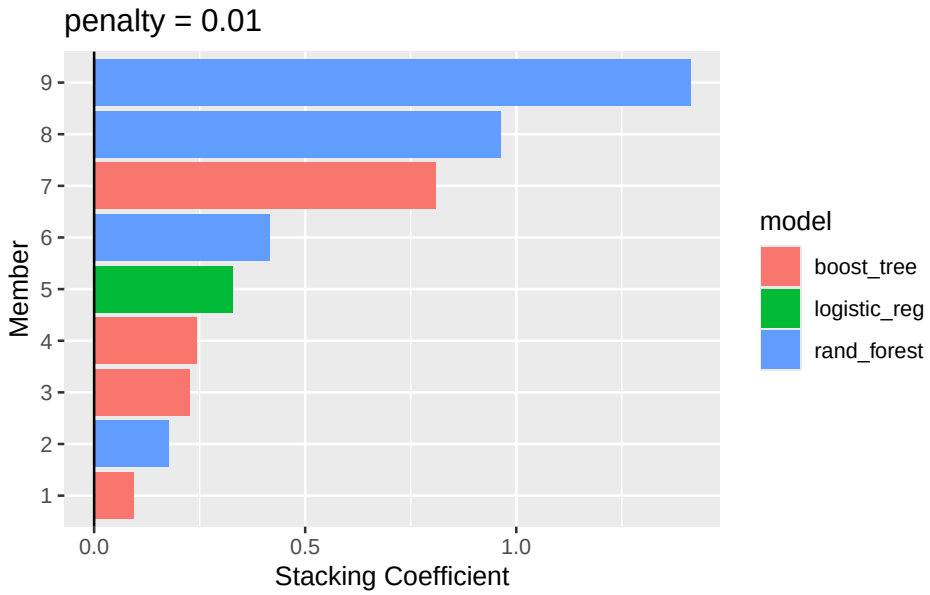
```
autoplot(model_stack_res)
```




```
autoplot(model_stack_res, type = "members")
```




```
autoplot(model_stack_res, type = "weights")
```




```
collect_parameters(model_stack_res, "logit_res")
```

```
# A tibble: 8 x 5
  member        penalty mixture terms                    coef
  <chr>           <dbl>   <dbl> <chr>                   <dbl>
1 logit_res_1_1  0.001      0   .pred_TRUE_logit_res_1_1 0.329
2 logit_res_1_2  0.0316     0   .pred_TRUE_logit_res_1_2 0
3 logit_res_1_3  1          0   .pred_TRUE_logit_res_1_3 0
4 logit_res_1_4  0.001      0.5 .pred_TRUE_logit_res_1_4 0
5 logit_res_1_5  0.0316     0.5 .pred_TRUE_logit_res_1_5 0
6 logit_res_1_6  1          0.5 .pred_TRUE_logit_res_1_6 0
7 logit_res_1_7  0.001      1   .pred_TRUE_logit_res_1_7 0
8 logit_res_1_8  0.0316     1   .pred_TRUE_logit_res_1_8 0
```

```
collect_parameters(model_stack_res, "bt_res")
```

```
# A tibble: 9 x 5
  member     tree_depth learn_rate terms                  coef
  <chr>           <int>      <dbl> <chr>                 <dbl>
1 bt_res_1_1          1     0.001  .pred_TRUE_bt_res_1_1 0
2 bt_res_1_2          2     0.001  .pred_TRUE_bt_res_1_2 0
3 bt_res_1_3          3     0.001  .pred_TRUE_bt_res_1_3 0
4 bt_res_1_4          1     0.0316 .pred_TRUE_bt_res_1_4 0
5 bt_res_1_5          2     0.0316 .pred_TRUE_bt_res_1_5 0
6 bt_res_1_6          3     0.0316 .pred_TRUE_bt_res_1_6 0.244
7 bt_res_1_7          1     1      .pred_TRUE_bt_res_1_7 0.810
8 bt_res_1_8          2     1      .pred_TRUE_bt_res_1_8 0.225
```

```
9 bt_res_1_9          3     1      .pred_TRUE_bt_res_1_9 0.0943
```

```
collect_parameters(model_stack_res, "rf_res")
```

```
# A tibble: 4 x 5
  member      mtry trees terms                  coef
  <chr>      <int> <int> <chr>                 <dbl>
1 rf_res_1_1     3   100 .pred_TRUE_rf_res_1_1 0.176
2 rf_res_1_2     3   300 .pred_TRUE_rf_res_1_2 1.41
3 rf_res_1_3     5   100 .pred_TRUE_rf_res_1_3 0.416
4 rf_res_1_4     5   300 .pred_TRUE_rf_res_1_4 0.963
```

0.1.1.3 Collect the parameters

```
#Final classification
stack_pred <- mimic_test |>
  bind_cols(
    predict(model_stack_res,
            new_data = mimic_test,
            type = "prob")) |>
  mutate(pred = .pred_FALSE < 0.5) |>
  mutate(pred = as.factor(pred)) |>
  print(width = Inf)
```

```
# A tibble: 47,223 x 23
   gender first_careunit insurance marital_status   race  age_intime creatinine
   <fct>  <fct>          <fct>     <fct>            <fct>      <dbl>      <dbl>
 1 1      2              2         5                5             52       0.7
 2 1      3              6         3                5             46       1.05
 3 1      1              3         4                4             57       0.9
 4 0      4              2         3.48443429415089 4             56       1.05
 5 1      2              3         3                5             83       1.4
 6 1      3              3         3                5             82       1.05
 7 1      2              3         5                5             81       0.6
 8 0      4              3         5                5             90       1.9
 9 0      4              6         4                5             53       0.9
10 1      1              6         3.48443429415089 5             58       1.05
   potassium sodium chloride bicarbonate hematocrit glucose   wbc heart_rate
       <dbl>  <dbl>    <dbl>       <dbl>      <dbl>   <dbl> <dbl>      <dbl>
 1      4.21   126       95          25         41.1    102   6.9        91
 2      4.1    139       98          24.1       34.9    125.  9.85       86
 3      3.5    137      102          24         34.9    125.  7.2        80
 4      4.21   138.     101.         18         34.3     95  16.8       110.
 5      4.21   138.     101.         26         22.4    133   9.8       114
 6      4.9    135       98          23         25.5    117  17.9        91
 7      4.4    144      111          27         34.7    173  10.5       106.
 8      4.4    140      102          23         29.9    105   5.1        93.5
```

```
 9      5.3    135      106         18         43.1    125. 16.9       106
10      4.21   138.     101.        24.1       34.9    125.  9.85       80
   `non-invasive_blood_pressure_systolic`
                                    <dbl>
 1                                   84
 2                                   73
 3                                   98.5
 4                                  112
 5                                  109
 6                                  118
 7                                  102
 8                                  108
 9                                  140
10                                  109
   `non-invasive_blood_pressure_diastolic` temperature_fahrenheit
                                     <dbl>                   <dbl>
 1                                      48                    98.7
 2                                      56                    97.7
 3                                      62                    97.2
 4                                      80                    97.9
 5                                      65                    97.7
 6                                      51                    96.9
 7                                      51                    98.6
 8                                      61                    98.1
 9                                      99                    96.7
10                                      72                    99
   respiratory_rate los_long .pred_FALSE .pred_TRUE pred
              <dbl> <fct>           <dbl>      <dbl> <fct>
 1             24   FALSE           0.480      0.520 TRUE
 2             19   FALSE           0.495      0.505 TRUE
 3             14   FALSE           0.554      0.446 FALSE
 4             21   TRUE            0.329      0.671 TRUE
 5             24   FALSE           0.436      0.564 TRUE
 6             18   TRUE            0.491      0.509 TRUE
 7             25   TRUE            0.466      0.534 TRUE
 8             22.5 TRUE            0.459      0.541 TRUE
 9             12   TRUE            0.560      0.440 FALSE
10             17   FALSE           0.357      0.643 TRUE
# i 47,213 more rows
```

```
yardstick::roc_auc(
  stack_pred,
  truth = los_long,
  contains(".pred_FALSE")
  )
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.621
```

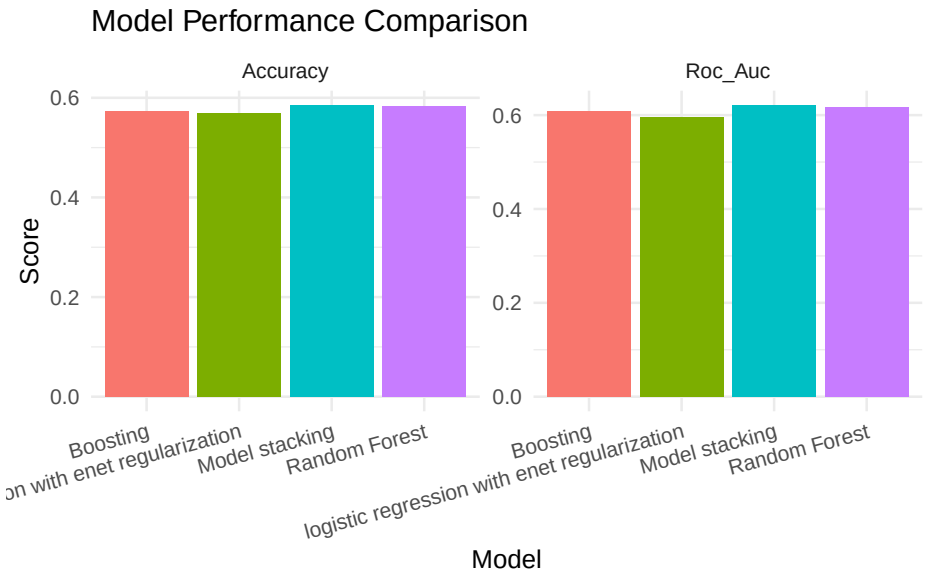
```
yardstick::accuracy(
  stack_pred,
  truth = los_long,
  estimate = pred
  )
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.585
```

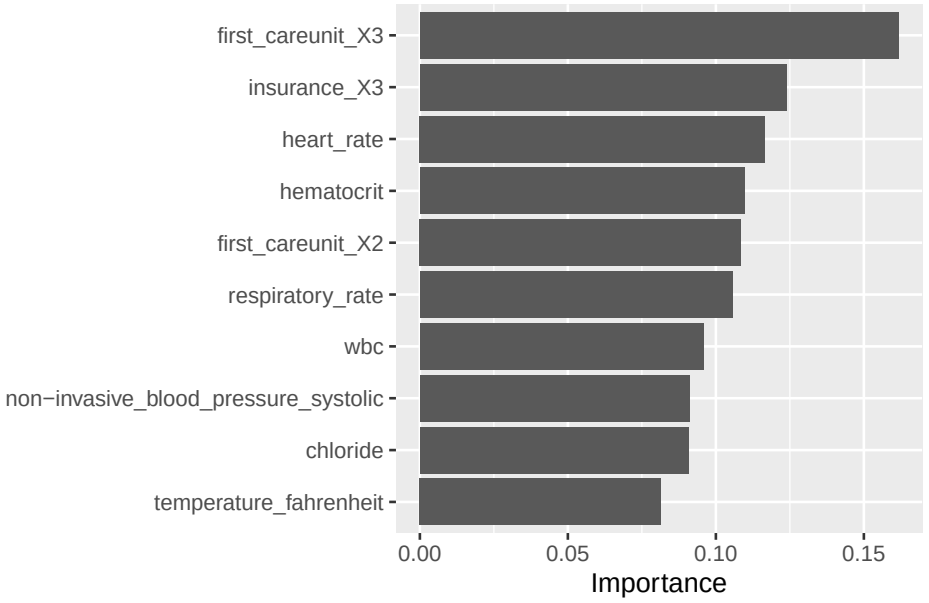
0.1.1.4 Compare the model results

Model	Roc_Auc	Accuracy
logistic regression with enet regularization	0.595	0.570
Boosting	0.609	0.574
Random Forest	0.617	0.583
Model stacking	0.621	0.585

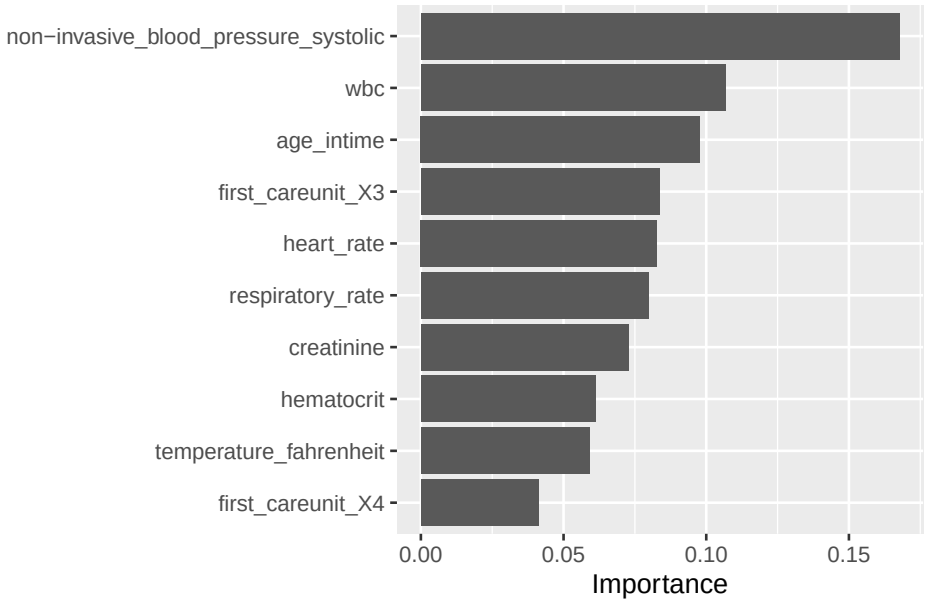
```
model_results <- data.frame(
  Model = c("logistic regression with enet regularization", "Boosting", "Random Forest", "Mo
  Roc_Auc = c(0.595, 0.609, 0.617, 0.621),
  Accuracy = c(0.570, 0.574, 0.583, 0.585)
)

model_results_long <- pivot_longer(model_results, cols = c(Roc_Auc, Accuracy),
                                   names_to = "Metric", values_to = "Value")

ggplot(model_results_long, aes(x = Model, y = Value, fill = Model)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Metric, scales = "free_y") +
  labs(title = "Model Performance Comparison", x = "Model", y = "Score") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 15, hjust = 1),
        legend.position = "none")
```


Model Performance Comparison
Accuracy
Roc_Auc
Score
0.0
0.2
0.4
0.6
Boosting
Model stacking
Random Forest
Model

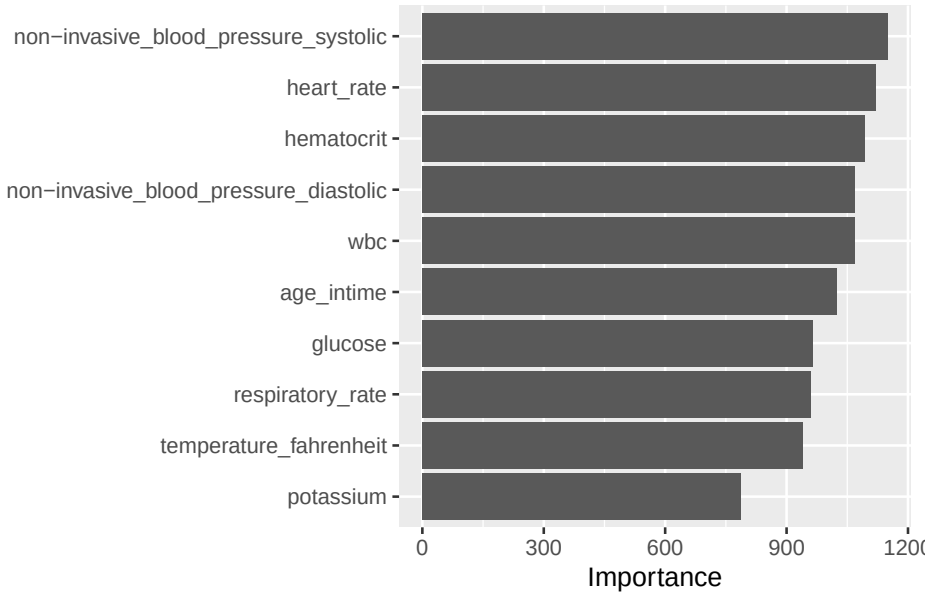
```
lg_vip
```


first_careunit_X3
insurance_X3
heart_rate
hematocrit
first_careunit_X2
respiratory_rate
wbc
non−invasive_blood_pressure_systolic
chloride
temperature_fahrenheit
0.00
0.05
0.10
0.15
Importance

```
bt_vip
```




```
rf_vip
```



Conclusion:

Model Stacking Achieves the Best Performance: The stacked model slightly outperforms all individual models, obtaining the highest ROC-AUC (0.621) and

accuracy (0.585). This suggests that combining multiple models leads to better generalization. Random Forest Performs Well: Among single models, Random Forest has the highest ROC-AUC (0.617) and accuracy (0.583), indicating that ensemble methods work well for this classification task.

Logistic Regression take First care unit as the most important feature. For Boosting and Random Forest, they both give more importance to Non-invasive blood pressure (systolic).

Logistic Regression is the most interpretable but performs the worst. Boosting and Random Forest offer a balance, but Random Forest achieves better accuracy. Stacked Models provide the best results but are the hardest to interpret.

Logistic regression is the easiest to explain because it assigns clear coefficients to features, making it easy to interpret. Boosting provides moderate interpretability because it provides feature importance, but relies on many decision trees, making it difficult to track a single prediction. Random forests have lower interpretability because they are composed of multiple trees, making it difficult to understand the specific decision path. Model stacking provides the highest performance but the most difficult to explain because it combines multiple models into a complex black box system. Ultimately, the trade-off between accuracy and interpretability depends on the use case - when interpretation is needed, logistic regression is more desirable, while ensemble models are more suitable for maximizing predictive performance.