Biostat 203B Homework 5

Due Mar 20 @ 11:59PM

Yanzi Sun 106183069

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.

```
#load necessary libraries
library(GGally)
```

```
Loading required package: ggplot2
```

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

```r
library(gtsummary)
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     1.0.2     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 becom
```

```r
library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.2.0 --
v broom        1.0.7     v rsample      1.2.1
v dials        1.3.0     v tune         1.2.1
v infer        1.0.7     v workflows    1.1.4
v modeldata    1.4.0     v workflowsets 1.1.0
v parsnip      1.2.1     v yardstick    1.3.2
v recipes      1.1.0
-- 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()
* Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(tensorflow)
library(keras)
```

```

Attaching package: 'keras'

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

    get_weights
```

```
library(reticulate)
library(xgboost)
```

```

Attaching package: 'xgboost'

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

    slice
```

```
library(ranger)
#load the mimic-icu dataset
mimic_icu_cohort <- readRDS("~/Desktop/203B/203B-HW/hw4/mimiciv_shiny/mimic_icu_cohort.rds")
  mutate(
    insurance = as.factor(insurance),
    marital_status = as.factor(marital_status),
    gender = as.factor(gender),
    los_long = as.logical(los_long), # Ensure it is TRUE/FALSE
    los_long = factor(los_long, levels = c(FALSE, TRUE),
                      labels = c("no", "yes"))) |>
    drop_na(los_long) #|>
    #slice_sample(prop = 0.25)  # downsample for faster computation

#have an overview of the dataset
#mimic_icu_cohort |> tbl_summary(by = los_long)
```

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.

```
##| eval: false
set.seed(203)

# sort
mimic_icu_cohort <- mimic_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id) |>
  select(first_careunit, gender, age_at_intime, marital_status, race,
         `heart rate`, `non invasive blood pressure systolic`,
         `non invasive blood pressure diastolic`,
         `respiratory rate`, `temperature fahrenheit`,
         bicarbonate, chloride, creatinine, glucose, potassium,
         sodium, hematocrit, wbc,
         los_long)

#Initial split into test and non-test sets

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

mimic_other <- training(data_split)
dim(mimic_other)
```

```
[1] 47221    19
```

```
mimic_test <- testing(data_split)
dim(mimic_test)
```

```
[1] 47223    19
```

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

Solution: I choose to do logistic regression, SVM, xgboost, and finally model stacking.

```
#this chunk is checking the data cleaniness, do not need to run so I commented
# mimic_other %>%
#   summarise(across(everything(), ~ sum(is.na(.)))) %>%
#    print(width = Inf)
#
# table(mimic_icu_cohort$los_long)
# summary(mimic_icu_cohort$los_long)
# str(mimic_icu_cohort$los_long)
# table(mimic_other$los_long)
```

##Here is the code for logistic regression:

```r
#Recipe
logit_recipe <-
  recipe(
    los_long ~ .,
    data = mimic_other
  ) |>
  step_unknown(all_nominal(), -all_outcomes()) |>  # Handle missing categorical values
  # mean imputation for numeric variable
  step_impute_mean(`heart rate`, `non invasive blood pressure systolic`, `non invasive blood
         `respiratory rate`, `temperature fahrenheit`,
         bicarbonate, chloride, creatinine, glucose, potassium, sodium,
         hematocrit, wbc) |>
  # mode imputation for catrgorical variable
  step_impute_mode(marital_status) |>
  # create traditional dummy variables
  step_dummy(all_nominal_predictors()) |>
  # zero-variance filter
  step_zv(all_numeric_predictors()) |>
  # center and scale numeric data
  step_normalize(all_numeric_predictors()) |>
  # estimate the means and standard deviations
  # prep(training = Heart_other, retain = TRUE) |>
  print()
```

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



-- Inputs

Number of variables by role

outcome:    1
predictor: 18



-- Operations
```

```
* Unknown factor level assignment for: all_nominal() -all_outcomes()

* Mean imputation for: `heart rate`, ...

* Mode imputation for: marital_status

* Dummy variables from: all_nominal_predictors()

* Zero variance filter on: all_numeric_predictors()

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

```
# Model
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
```

```
# Workflow, bundle the recipe and model

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

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

-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

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

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
# Tune the model(the penalty and mixture hyperparameters).

param_grid <- grid_regular(
  penalty(range = c(-6, 3)),
  mixture(),
  levels = c(100, 5)
  ) |>
  print()
```

```
# A tibble: 500 x 2
      penalty mixture
        <dbl>   <dbl>
 1 0.000001         0
 2 0.00000123       0
 3 0.00000152       0
 4 0.00000187       0
 5 0.00000231       0
 6 0.00000285       0
 7 0.00000351       0
 8 0.00000433       0
 9 0.00000534       0
10 0.00000658       0
# i 490 more rows
```

```
# Set Cross-Validation partitions
set.seed(203)

folds <- vfold_cv(mimic_other, 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
```

```
# fit cross-validation
(logit_fit <- logit_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid,
    metrics = metric_set(roc_auc, accuracy)
    )) |>
  system.time()
```

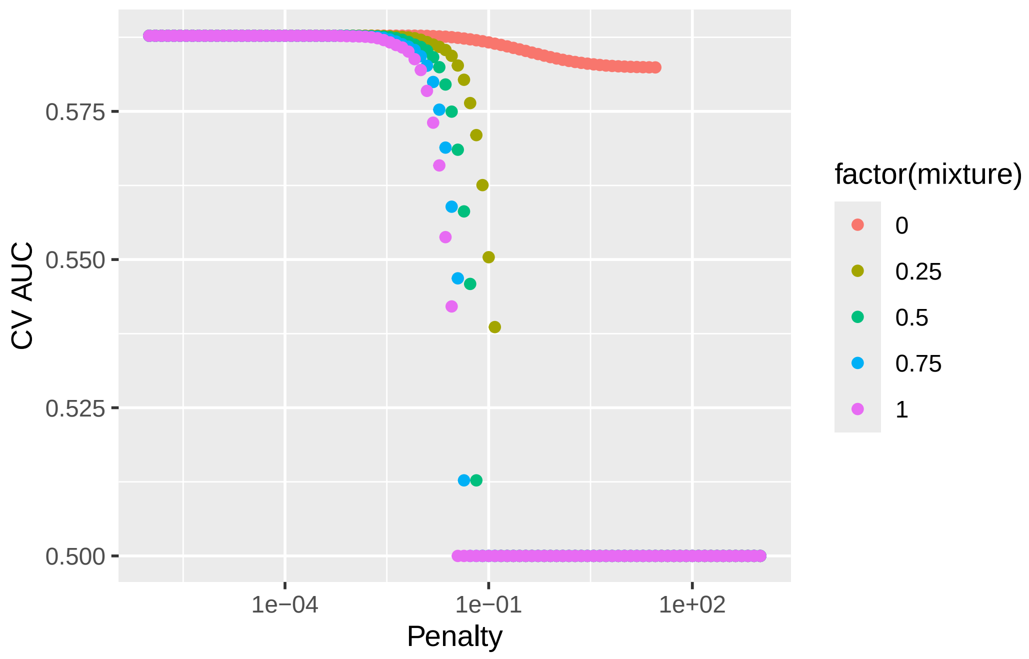
```
   user  system elapsed
 29.180   2.178  31.417
```

```
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 [1,000 x 6]> <tibble [0 x 3]>
2 <split [37777/9444]> Fold2 <tibble [1,000 x 6]> <tibble [0 x 3]>
3 <split [37777/9444]> Fold3 <tibble [1,000 x 6]> <tibble [0 x 3]>
4 <split [37777/9444]> Fold4 <tibble [1,000 x 6]> <tibble [0 x 3]>
5 <split [37777/9444]> Fold5 <tibble [1,000 x 6]> <tibble [0 x 3]>
```

```r
# Visualize
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: 1,000 x 8
      penalty mixture .metric  .estimator  mean     n std_err
        <dbl>   <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
 1 0.000001         0 accuracy binary     0.562     5 0.00230
 2 0.000001         0 roc_auc  binary     0.588     5 0.00246
 3 0.00000123       0 accuracy binary     0.562     5 0.00230
 4 0.00000123       0 roc_auc  binary     0.588     5 0.00246
 5 0.00000152       0 accuracy binary     0.562     5 0.00230
 6 0.00000152       0 roc_auc  binary     0.588     5 0.00246
 7 0.00000187       0 accuracy binary     0.562     5 0.00230
 8 0.00000187       0 roc_auc  binary     0.588     5 0.00246
 9 0.00000231       0 accuracy binary     0.562     5 0.00230
10 0.00000231       0 roc_auc  binary     0.588     5 0.00246
   .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 990 more rows
```



```
#show top 5 models and select the best one
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 0.00811          0 roc_auc binary     0.588     5 0.00246 Preprocessor1_Model~
2 0.000001         0 roc_auc binary     0.588     5 0.00246 Preprocessor1_Model~
3 0.00000123       0 roc_auc binary     0.588     5 0.00246 Preprocessor1_Model~
4 0.00000152       0 roc_auc binary     0.588     5 0.00246 Preprocessor1_Model~
5 0.00000187       0 roc_auc binary     0.588     5 0.00246 Preprocessor1_Model~
```

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

```
# A tibble: 1 x 3
  penalty mixture .config
    <dbl>   <dbl> <chr>
1 0.00811       0 Preprocessor1_Model044
```

```
#finalize workflow
# Final workflow
final_wf <- logit_wf |>
  finalize_workflow(best_logit)
final_wf
```

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

-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

Main Arguments:
  penalty = 0.00811130830789687
  mixture = 0

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

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

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits                id              .metrics .notes   .predictions .workflow
  <list>                <chr>           <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp~ <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
final_fit |>
  collect_metrics()
```

```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.561 Preprocessor1_Model1
2 roc_auc     binary         0.585 Preprocessor1_Model1
3 brier_class binary         0.245 Preprocessor1_Model1
```

##Here is the code for SVM:

```
# Recipe
library(kernlab)
```

```
Attaching package: 'kernlab'


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

    alpha


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

    cross


The following object is masked from 'package:ggplot2':

    alpha
```

```r
svm_recipe <-
  recipe(
    los_long ~ .,
    data = mimic_other
  ) |>
  step_unknown(all_nominal(), -all_outcomes()) |>  # Handle missing categorical values
  # mean imputation for numeric variable
  step_impute_mean(`heart rate`, `non invasive blood pressure systolic`, `non invasive blood
         `respiratory rate`, `temperature fahrenheit`,
         bicarbonate, chloride, creatinine, glucose, potassium, sodium,
         hematocrit, wbc) |>
  # mode imputation for catrgorical variable
  step_impute_mode(marital_status) |>
  # create traditional dummy variables
  step_dummy(all_nominal_predictors()) |>
  # zero-variance filter
  step_zv(all_numeric_predictors()) |>
  # center and scale numeric data
  step_normalize(all_numeric_predictors()) |>
  # estimate the means and standard deviations
  # prep(training = Heart_other, retain = TRUE) |>
  print()
```

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



-- Inputs

Number of variables by role

outcome:    1
predictor: 18



-- Operations
```

```
* Unknown factor level assignment for: all_nominal() -all_outcomes()

* Mean imputation for: `heart rate`, ...

* Mode imputation for: marital_status

* Dummy variables from: all_nominal_predictors()

* Zero variance filter on: all_numeric_predictors()

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

```r
# Model
svm_mod <-
  svm_rbf(
    mode = "classification",
    cost = tune(),
    rbf_sigma = tune()
  ) |>
  set_engine("kernlab")
svm_mod
```

```
Radial Basis Function Support Vector Machine Model Specification (classification)

Main Arguments:
  cost = tune()
  rbf_sigma = tune()

Computational engine: kernlab
```

```r
# Bundle recipe & Model
svm_wf <- workflow() |>
  add_recipe(svm_recipe) |>
  add_model(svm_mod)
svm_wf
```

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

```
-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

-- Model -----------------------------------------------------------------------
Radial Basis Function Support Vector Machine Model Specification (classification)

Main Arguments:
  cost = tune()
  rbf_sigma = tune()

Computational engine: kernlab
```

```
# Tune the model
 # Trail #1 with best roc-auc ~0.593, since the best point is on the top-right edge, tune the
 # For Trail #2, reduced the sample size to 1/4 of original size to speed up the computation
param_grid2 <- grid_regular(
  cost(range = c(0, 6)),
  rbf_sigma(range = c(-4, -1)),
  levels = c(3, 3)
  )
# param_grid1 <- grid_regular(
#   cost(range = c(-8, 5)),
#   rbf_sigma(range = c(-5, -3)),
#   levels = c(3, 3)
#   )
param_grid2
```

```
# A tibble: 9 x 2
   cost rbf_sigma
  <dbl>     <dbl>
1     1   0.0001
2     8   0.0001
3    64   0.0001
4     1   0.00316
```

```
5     8   0.00316
6    64   0.00316
7     1   0.1
8     8   0.1
9    64   0.1
```

```
# CV, since SVM is slow, I choose to use only 2 folds here.
set.seed(203)

folds <- vfold_cv(mimic_other, v = 2)
folds
```

```
#  2-fold cross-validation
# A tibble: 2 x 2
  splits                id
  <list>                <chr>
1 <split [23610/23611]> Fold1
2 <split [23611/23610]> Fold2
```

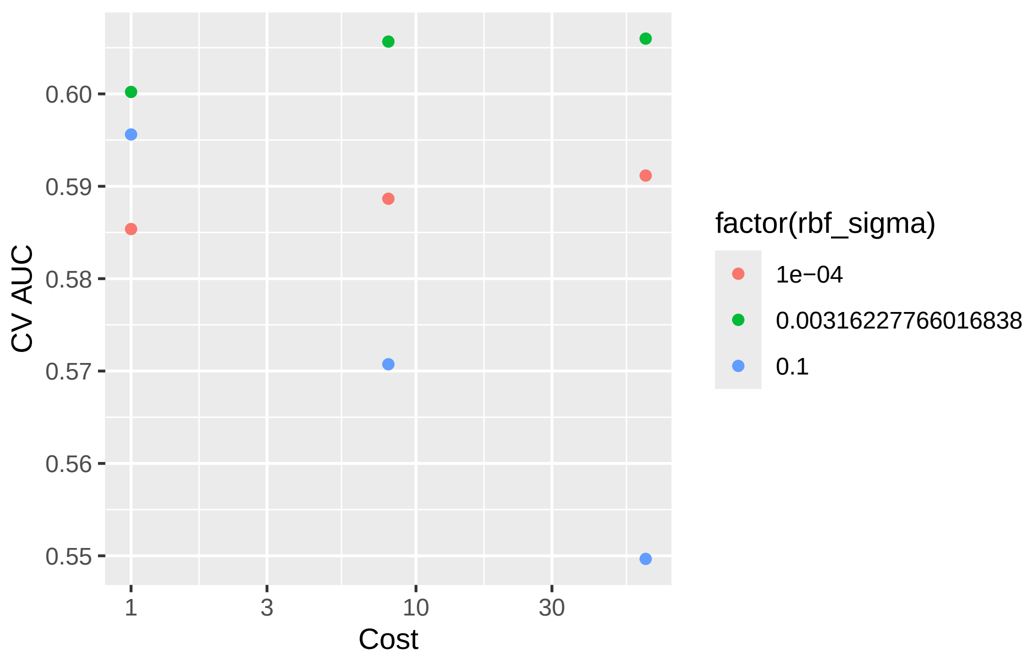
```
# Fit cross-validation
svm_fit <- svm_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid2,
    metrics = metric_set(roc_auc, accuracy)
    )
svm_fit
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 x 4
  splits                id    .metrics          .notes
  <list>                <chr> <list>            <list>
1 <split [23610/23611]> Fold1 <tibble [18 x 6]> <tibble [0 x 3]>
2 <split [23611/23610]> Fold2 <tibble [18 x 6]> <tibble [0 x 3]>
```

```
# Visualize CV results
svm_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
```

```
ggplot(mapping = aes(x = cost, y = mean, color = factor(rbf_sigma))) +
geom_point() +
labs(x = "Cost", y = "CV AUC") +
scale_x_log10()
```

```
# A tibble: 18 x 8
    cost rbf_sigma .metric  .estimator  mean     n  std_err .config
   <dbl>     <dbl> <chr>    <chr>      <dbl> <int>    <dbl> <chr>
 1     1    0.0001  accuracy binary     0.548     2 0.00154  Preprocessor1_Model1
 2     1    0.0001  roc_auc  binary     0.585     2 0.00620  Preprocessor1_Model1
 3     8    0.0001  accuracy binary     0.559     2 0.00340  Preprocessor1_Model2
 4     8    0.0001  roc_auc  binary     0.589     2 0.00551  Preprocessor1_Model2
 5    64    0.0001  accuracy binary     0.563     2 0.00414  Preprocessor1_Model3
 6    64    0.0001  roc_auc  binary     0.591     2 0.00591  Preprocessor1_Model3
 7     1    0.00316 accuracy binary     0.569     2 0.00372  Preprocessor1_Model4
 8     1    0.00316 roc_auc  binary     0.600     2 0.00623  Preprocessor1_Model4
 9     8    0.00316 accuracy binary     0.575     2 0.00484  Preprocessor1_Model5
10     8    0.00316 roc_auc  binary     0.606     2 0.00643  Preprocessor1_Model5
11    64    0.00316 accuracy binary     0.575     2 0.00376  Preprocessor1_Model6
12    64    0.00316 roc_auc  binary     0.606     2 0.00519  Preprocessor1_Model6
13     1    0.1     accuracy binary     0.568     2 0.00219  Preprocessor1_Model7
14     1    0.1     roc_auc  binary     0.596     2 0.00387  Preprocessor1_Model7
15     8    0.1     accuracy binary     0.553     2 0.00224  Preprocessor1_Model8
16     8    0.1     roc_auc  binary     0.571     2 0.00128  Preprocessor1_Model8
17    64    0.1     accuracy binary     0.536     2 0.000837 Preprocessor1_Model9
18    64    0.1     roc_auc  binary     0.550     2 0.000656 Preprocessor1_Model9
```



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

```
# A tibble: 5 x 8
   cost rbf_sigma .metric .estimator  mean     n std_err .config
  <dbl>     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1    64   0.00316 roc_auc binary     0.606     2 0.00519 Preprocessor1_Model6
2     8   0.00316 roc_auc binary     0.606     2 0.00643 Preprocessor1_Model5
3     1   0.00316 roc_auc binary     0.600     2 0.00623 Preprocessor1_Model4
4     1   0.1     roc_auc binary     0.596     2 0.00387 Preprocessor1_Model7
5    64   0.0001  roc_auc binary     0.591     2 0.00591 Preprocessor1_Model3
```

```
best_svm <- svm_fit |>
  select_best(metric = "roc_auc")
best_svm
```

```
# A tibble: 1 x 3
   cost rbf_sigma .config
  <dbl>     <dbl> <chr>
1    64   0.00316 Preprocessor1_Model6
```

```r
# Final workflow
final_wf <- svm_wf |>
  finalize_workflow(best_svm)
final_wf
```

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

-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

-- Model -----------------------------------------------------------------------
Radial Basis Function Support Vector Machine Model Specification (classification)

Main Arguments:
  cost = 64
  rbf_sigma = 0.00316227766016838

Computational engine: kernlab
```

```r
# Fit the whole training set, then predict the test cases
final_fit <-
  final_wf |>
  last_fit(data_split)
final_fit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits                id               .metrics .notes   .predictions .workflow
  <list>                <chr>            <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp~ <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
final_fit |>
  collect_metrics()
```

```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.579 Preprocessor1_Model1
2 roc_auc     binary         0.610 Preprocessor1_Model1
3 brier_class binary         0.241 Preprocessor1_Model1
```

##Here is the code for xgBoost:

```
# Recipe
gb_recipe <-
  recipe(
    los_long ~ .,
    data = mimic_other
  ) |>
  step_unknown(all_nominal(), -all_outcomes()) |>  # Handle missing categorical values
  # mean imputation for numeric variable
  step_impute_mean(`heart rate`, `non invasive blood pressure systolic`, `non invasive blood
         `respiratory rate`, `temperature fahrenheit`,
         bicarbonate, chloride, creatinine, glucose, potassium,
         sodium, hematocrit, wbc) |>
  # mode imputation for catrgorical variable
  step_impute_mode(marital_status) |>
  # create traditional dummy variables
  step_dummy(all_nominal_predictors()) |>
  # zero-variance filter
  step_zv(all_numeric_predictors()) |>
  # center and scale numeric data
  step_normalize(all_numeric_predictors()) |>
  # estimate the means and standard deviations
  # prep(training = Heart_other, retain = TRUE) |>
  print()
```

```
-- Recipe ----------------------------------------------------------------------
-- Inputs

Number of variables by role

outcome:    1
predictor: 18



-- Operations

* Unknown factor level assignment for: all_nominal() -all_outcomes()

* Mean imputation for: `heart rate`, ...

* Mode imputation for: marital_status

* Dummy variables from: all_nominal_predictors()

* Zero variance filter on: all_numeric_predictors()

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

```r
# Model
gb_mod <-
  boost_tree(
    mode = "classification",
    trees = 1000,
    tree_depth = tune(),
    learn_rate = tune()
  ) |>
  set_engine("xgboost")
gb_mod
```

```
Boosted Tree Model Specification (classification)

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

Computational engine: xgboost
```

```r
# bundle recipe and model(workflow step)
gb_wf <- workflow() |>
  add_recipe(gb_recipe) |>
  add_model(gb_mod)
gb_wf
```

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

-- Preprocessor ------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

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

Computational engine: xgboost
```

```r
# Tune
param_grid <- grid_regular(
  tree_depth(range = c(1L, 3L)),
  learn_rate(range = c(-5, 2), trans = log10_trans()),
  levels = c(3, 3)
  )
param_grid
```

```
# A tibble: 9 x 2
  tree_depth learn_rate
       <int>      <dbl>
1          1    0.00001
2          2    0.00001
3          3    0.00001
4          1    0.0316
5          2    0.0316
6          3    0.0316
7          1  100
8          2  100
9          3  100
```

```r
#CV
set.seed(203)

folds <- vfold_cv(mimic_other, v = 2)
folds
```

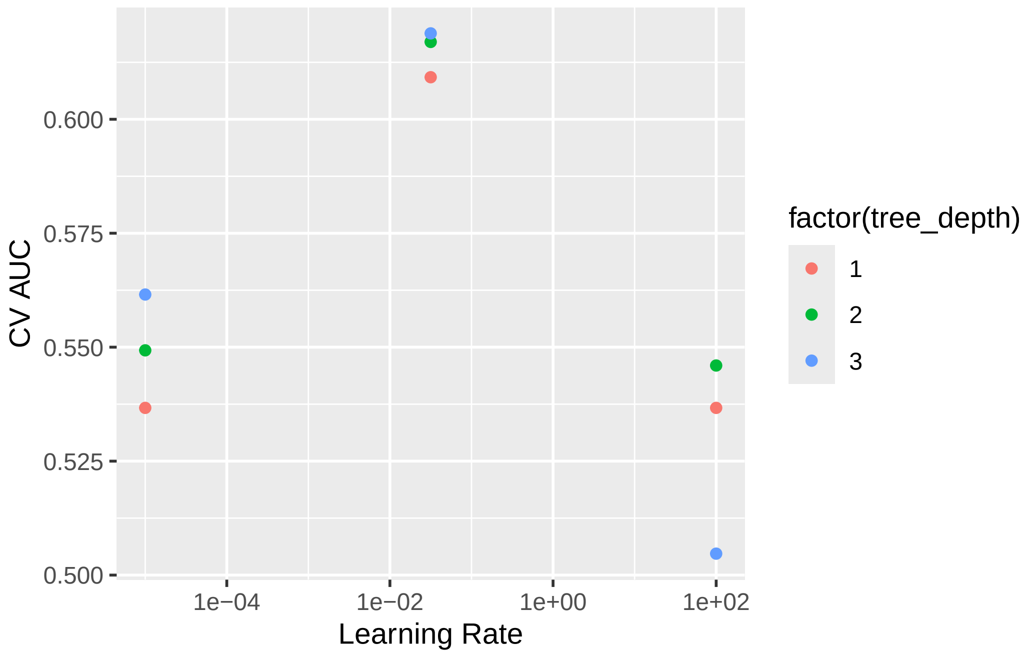
```
#  2-fold cross-validation
# A tibble: 2 x 2
  splits                id
  <list>                <chr>
1 <split [23610/23611]> Fold1
2 <split [23611/23610]> Fold2
```

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

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 x 4
  splits                id    .metrics          .notes
  <list>                <chr> <list>            <list>
1 <split [23610/23611]> Fold1 <tibble [18 x 6]> <tibble [0 x 3]>
2 <split [23611/23610]> Fold2 <tibble [18 x 6]> <tibble [0 x 3]>
```

```
# Visualize
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: 18 x 8
   tree_depth learn_rate .metric  .estimator  mean     n  std_err
        <int>      <dbl> <chr>    <chr>      <dbl> <int>    <dbl>
 1          1    0.00001 accuracy binary     0.537     2 0.00482
 2          1    0.00001 roc_auc  binary     0.537     2 0.00383
 3          2    0.00001 accuracy binary     0.545     2 0.00200
 4          2    0.00001 roc_auc  binary     0.549     2 0.000879
 5          3    0.00001 accuracy binary     0.547     2 0.00440
 6          3    0.00001 roc_auc  binary     0.562     2 0.000426
 7          1    0.0316  accuracy binary     0.575     2 0.00465
 8          1    0.0316  roc_auc  binary     0.609     2 0.00469
 9          2    0.0316  accuracy binary     0.581     2 0.00459
10          2    0.0316  roc_auc  binary     0.617     2 0.00403
11          3    0.0316  accuracy binary     0.585     2 0.00548
12          3    0.0316  roc_auc  binary     0.619     2 0.00413
13          1  100       accuracy binary     0.537     2 0.00482
14          1  100       roc_auc  binary     0.537     2 0.00383
15          2  100       accuracy binary     0.545     2 0.00200
16          2  100       roc_auc  binary     0.546     2 0.00421
17          3  100       accuracy binary     0.500     2 0.0425
18          3  100       roc_auc  binary     0.505     2 0.0497
   .config
   <chr>
 1 Preprocessor1_Model1
 2 Preprocessor1_Model1
 3 Preprocessor1_Model2
 4 Preprocessor1_Model2
 5 Preprocessor1_Model3
 6 Preprocessor1_Model3
 7 Preprocessor1_Model4
 8 Preprocessor1_Model4
 9 Preprocessor1_Model5
10 Preprocessor1_Model5
11 Preprocessor1_Model6
12 Preprocessor1_Model6
13 Preprocessor1_Model7
14 Preprocessor1_Model7
15 Preprocessor1_Model8
16 Preprocessor1_Model8
17 Preprocessor1_Model9
18 Preprocessor1_Model9
```



```r
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    0.0316  roc_auc binary     0.619     2 0.00413  Preprocessor1_M~
2           2    0.0316  roc_auc binary     0.617     2 0.00403  Preprocessor1_M~
3           1    0.0316  roc_auc binary     0.609     2 0.00469  Preprocessor1_M~
4           3    0.00001 roc_auc binary     0.562     2 0.000426 Preprocessor1_M~
5           2    0.00001 roc_auc binary     0.549     2 0.000879 Preprocessor1_M~
```

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

```
# A tibble: 1 x 3
  tree_depth learn_rate .config
       <int>      <dbl> <chr>
1          3     0.0316 Preprocessor1_Model6
```

```
#finalize model
final_wf <- gb_wf |>
  finalize_workflow(best_gb)
final_wf
```

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

-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

Main Arguments:
  trees = 1000
  tree_depth = 3
  learn_rate = 0.0316227766016838

Computational engine: xgboost
```

```
final_fit <-
  final_wf |>
  last_fit(data_split)
final_fit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits                id              .metrics .notes   .predictions .workflow
  <list>                <chr>           <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp~ <tibble> <tibble> <tibble>     <workflow>
```

```
final_fit |> collect_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         0.622 Preprocessor1_Model1
3 brier_class binary         0.238 Preprocessor1_Model1
```

##Model Stacking code:

```
# set up 2-fold CV
set.seed(203)
library(stacks)
folds <- vfold_cv(mimic_other, v = 2)

# Base models
# logistic regression
logit_mod <-
  logistic_reg(
    penalty = tune(),
    mixture = tune()
  ) |>
  set_engine("glmnet", standardize = FALSE)
logit_mod
```

```
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(gb_recipe) |>
  add_model(logit_mod)
logit_wf
```

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

-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

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

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
logit_grid <- grid_regular(
  penalty(range = c(-6, 3)),
  mixture(),
  levels = c(100, 5)
  )

logit_res <- logit_wf |>
  tune_grid(
    resamples = folds,
    grid = logit_grid,
    control = control_stack_grid()
    )
```

```
i The workflow being saved contains a recipe, which is 5.85 Mb in i memory. If
this was not intentional, please set the control setting i `save_workflow =
FALSE`.
```

```r
logit_res
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 x 5
  splits                id    .metrics              .notes           .predictions
  <list>                <chr> <list>                <list>           <list>
1 <split [23610/23611]> Fold1 <tibble [1,500 x 6]> <tibble [0 x 3]> <tibble>
2 <split [23611/23610]> Fold2 <tibble [1,500 x 6]> <tibble [0 x 3]> <tibble>
```

```r
#random forest
library(ranger)
rf_mod <-
  rand_forest(
    mode = "classification",
    mtry = tune(),
    trees = tune()
    ) |>
  set_engine("ranger",
    importance = "impurity")
rf_mod
```

```
Random Forest Model Specification (classification)
```

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

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```
#i used xgboost recipe here since they are essentially the same
rf_wf <- workflow() |>
  add_recipe(gb_recipe) |>
  add_model(rf_mod)
rf_wf
```

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

-- Preprocessor ----------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

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

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```r
rf_grid <- grid_regular(
  trees(range = c(100L,500L)),
  mtry(range = c(1L, 5L)),
  levels = c(5,5)
)

rf_res <- tune_grid(
  object = rf_wf,
  resamples = folds,
  grid = rf_grid,
  control = control_stack_grid()
)
```

```
i The workflow being saved contains a recipe, which is 5.85 Mb in i memory. If
this was not intentional, please set the control setting i `save_workflow =
FALSE`.
```

```r
rf_res
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 x 5
  splits                id    .metrics          .notes           .predictions
  <list>                <chr> <list>            <list>           <list>
1 <split [23610/23611]> Fold1 <tibble [75 x 6]> <tibble [0 x 3]> <tibble>
2 <split [23611/23610]> Fold2 <tibble [75 x 6]> <tibble [0 x 3]> <tibble>
```

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

```
Boosted Tree Model Specification (classification)

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

Computational engine: xgboost
```

```r
gb_wf <- workflow() |>
      add_recipe(gb_recipe) |>
      add_model(gb_mod)
gb_wf
```

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

-- Preprocessor ------------------------------------------------------------
6 Recipe Steps

* step_unknown()
* step_impute_mean()
* step_impute_mode()
* step_dummy()
* step_zv()
* step_normalize()

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

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

Computational engine: xgboost
```

```r
gb_grid <- grid_regular(
  tree_depth(range = c(1L, 3L)),
  learn_rate(range = c(-5, 2), trans = log10_trans()),
  levels = c(3, 3)
  )
gb_grid
```

```
# A tibble: 9 x 2
```

```
  tree_depth learn_rate
       <int>      <dbl>
1          1    0.00001
2          2    0.00001
3          3    0.00001
4          1    0.0316
5          2    0.0316
6          3    0.0316
7          1  100
8          2  100
9          3  100
```

```
gb_res <-
  tune_grid(
    object = gb_wf,
    resamples = folds,
    grid = gb_grid,
    control = control_stack_grid()
  )
```

```
i The workflow being saved contains a recipe, which is 5.85 Mb in i memory. If
this was not intentional, please set the control setting i `save_workflow =
FALSE`.
```

```
gb_res
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 x 5
  splits                id    .metrics          .notes           .predictions
  <list>                <chr> <list>            <list>           <list>
1 <split [23610/23611]> Fold1 <tibble [27 x 6]> <tibble [0 x 3]> <tibble>
2 <split [23611/23610]> Fold2 <tibble [27 x 6]> <tibble [0 x 3]> <tibble>
```

```
#stacking models
mimic_model_st <- stacks() |>
  add_candidates(logit_res)|>
  add_candidates(rf_res) |>
  add_candidates(gb_res) |>
  blend_predictions(
    penalty = 10^(-6:2),
    metrics = c("roc_auc")
  ) |>
  fit_members()
```

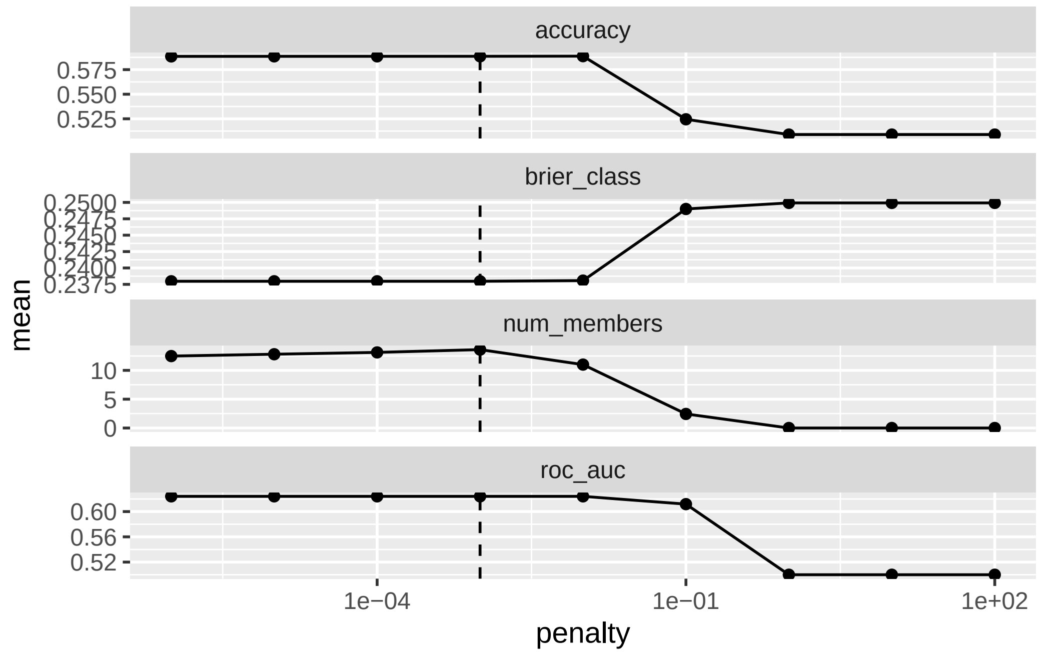
```
Warning: Predictions from 742 candidates were identical to those from existing
candidates and were removed from the data stack.
```

```
Warning: The `...` are not used in this function but one or more arguments were
passed: 'metrics'
```

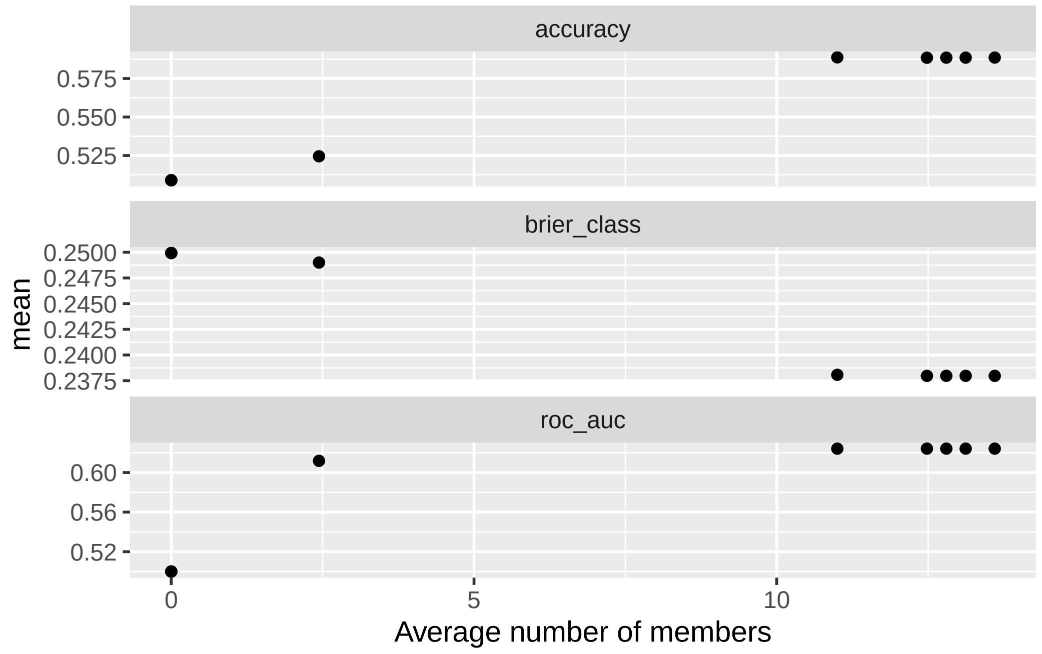
```
mimic_model_st
```

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


Out of 163 possible candidate members, the ensemble retained 15.

Penalty: 0.001.

Mixture: 1.


The 10 highest weighted member classes are:

# A tibble: 10 x 3
   member                    type          weight
   <chr>                     <chr>          <dbl>
 1 .pred_yes_gb_res_1_6      boost_tree    1.83
 2 .pred_yes_rf_res_1_08     rand_forest   0.825
 3 .pred_yes_rf_res_1_20     rand_forest   0.601
 4 .pred_yes_rf_res_1_17     rand_forest   0.452
 5 .pred_yes_rf_res_1_07     rand_forest   0.408
 6 .pred_yes_rf_res_1_21     rand_forest   0.348
 7 .pred_yes_rf_res_1_06     rand_forest   0.223
 8 .pred_yes_logit_res_1_101 logistic_reg 0.118
 9 .pred_yes_logit_res_1_201 logistic_reg 0.0928
10 .pred_yes_rf_res_1_09     rand_forest   0.0871
```

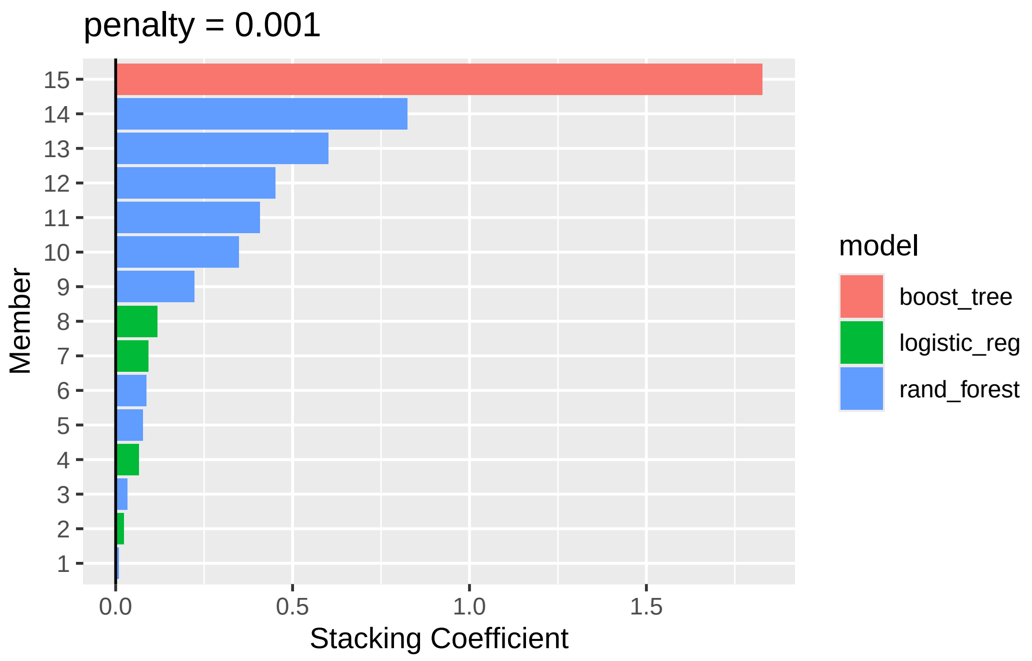
```
autoplot(mimic_model_st)
```



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



```r
autoplot(mimic_model_st, type = "weights")
```



```r
#collect_metrics(mimic_model_st, "rf_res")
```

```r
collect_parameters(mimic_model_st, "rf_res")
```

```
# A tibble: 25 x 5
   member       mtry trees terms                      coef
   <chr>       <int> <int> <chr>                     <dbl>
 1 rf_res_1_01     1   100 .pred_yes_rf_res_1_01 0
 2 rf_res_1_02     1   200 .pred_yes_rf_res_1_02 0
 3 rf_res_1_03     1   300 .pred_yes_rf_res_1_03 0
 4 rf_res_1_04     1   400 .pred_yes_rf_res_1_04 0
 5 rf_res_1_05     1   500 .pred_yes_rf_res_1_05 0
 6 rf_res_1_06     2   100 .pred_yes_rf_res_1_06 0.223
 7 rf_res_1_07     2   200 .pred_yes_rf_res_1_07 0.408
 8 rf_res_1_08     2   300 .pred_yes_rf_res_1_08 0.825
 9 rf_res_1_09     2   400 .pred_yes_rf_res_1_09 0.0871
10 rf_res_1_10     2   500 .pred_yes_rf_res_1_10 0
# i 15 more rows
```

```r
#final classification
mimic_pred <- mimic_test %>%
  bind_cols(predict(mimic_model_st, ., type = "prob")) %>%
  print(width = Inf)
```

```
# A tibble: 47,223 x 21
   first_careunit                                   gender age_at_intime
   <fct>                                            <fct>          <int>
 1 Medical Intensive Care Unit (MICU)               F                 52
 2 Medical/Surgical Intensive Care Unit (MICU/SICU) F                 46
 3 Cardiac Vascular Intensive Care Unit (CVICU)     F                 57
 4 Other                                            M                 56
 5 Medical Intensive Care Unit (MICU)               F                 83
 6 Medical/Surgical Intensive Care Unit (MICU/SICU) F                 82
 7 Medical Intensive Care Unit (MICU)               F                 81
 8 Other                                            M                 90
 9 Other                                            M                 53
10 Cardiac Vascular Intensive Care Unit (CVICU)     F                 58
   marital_status race  `heart rate` `non invasive blood pressure systolic`
   <fct>          <fct>        <dbl>                                  <dbl>
 1 WIDOWED        WHITE           91                                     84
 2 MARRIED        WHITE           86                                     73
 3 SINGLE         Other           80                                    104
 4 <NA>           Other          111                                    112
 5 MARRIED        WHITE           71                                    126
 6 MARRIED        WHITE           71                                    126
 7 WIDOWED        WHITE          124                                     87
 8 WIDOWED        WHITE           96                                    108
 9 SINGLE         WHITE          106                                    140
10 <NA>           WHITE           80                                    109
   `non invasive blood pressure diastolic` `respiratory rate`
                                     <dbl>              <dbl>
 1                                      48                 24
 2                                      56                 19
 3                                      70                 14
 4                                      80                 22
 5                                      61                 18
 6                                      61                 18
 7                                      42                 25
 8                                      61                 26
 9                                      99                 12
10                                      72                 17
```

```
   `temperature fahrenheit` bicarbonate chloride creatinine glucose potassium
                      <dbl>       <dbl>    <dbl>      <dbl>   <dbl>     <dbl>
 1                     98.7          25       95        0.7     102       6.7
 2                     97.7          NA       98       NA        NA       4.1
 3                     97.2          24      102        0.9     288       3.5
 4                     97.9          18       NA        3.1      95       6.5
 5                     95.9          26       85        1.4     133       5.7
 6                     95.9          23       98        2.8     117       4.9
 7                    103.           27      111        0.6     173       4.4
 8                     98.1          23      102        1.9     105       4.4
 9                     96.7          18      106        0.9     269       5.3
10                     99            NA       NA       NA        NA      NA
   sodium hematocrit   wbc los_long .pred_no .pred_yes
    <dbl>      <dbl> <dbl> <fct>       <dbl>     <dbl>
 1    126       41.1   6.9 no          0.520     0.480
 2    139       NA    NA   no          0.492     0.508
 3    137       34.9   7.2 no          0.614     0.386
 4    125       34.3  16.8 yes         0.359     0.641
 5    120       22.4   9.8 no          0.548     0.452
 6    135       25.5  17.9 yes         0.556     0.444
 7    144       34.7  10.5 yes         0.382     0.618
 8    140       29.9   5.1 yes         0.473     0.527
 9    135       43.1  16.9 yes         0.485     0.515
10     NA       NA    NA   no          0.323     0.677
# i 47,213 more rows
```

```
yardstick::roc_auc(
  mimic_pred,
  truth = los_long,
  contains(".pred_No")
)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.629
```

```
mimic_pred <- mimic_test |>
  select(los_long) |>
  bind_cols(
    predict(mimic_model_st,
            mimic_test,
            type = "class",
            members = TRUE)) |>
  print(width = Inf)
```

```
# A tibble: 47,223 x 17
   los_long .pred_class .pred_class_logit_res_1_001 .pred_class_logit_res_1_101
   <fct>    <fct>       <fct>                       <fct>
 1 no       no          no                          no
 2 no       yes         no                          no
 3 no       no          no                          no
 4 yes      yes         yes                         yes
 5 no       no          no                          no
 6 yes      no          no                          no
 7 yes      yes         yes                         yes
 8 yes      yes         yes                         yes
 9 yes      yes         no                          no
10 no       yes         yes                         yes
   .pred_class_logit_res_1_201 .pred_class_logit_res_1_301
   <fct>                       <fct>
 1 no                          no
 2 no                          no
 3 no                          no
 4 yes                         yes
 5 no                          no
 6 no                          no
 7 yes                         yes
 8 yes                         yes
 9 no                          no
10 yes                         yes
   .pred_class_rf_res_1_06 .pred_class_rf_res_1_07 .pred_class_rf_res_1_08
   <fct>                   <fct>                   <fct>
 1 no                      no                      no
 2 yes                     yes                     yes
 3 no                      no                      no
 4 yes                     yes                     yes
 5 no                      no                      no
 6 no                      no                      no
 7 yes                     yes                     yes
 8 no                      yes                     no
 9 no                      no                      no
10 yes                     yes                     yes
   .pred_class_rf_res_1_09 .pred_class_rf_res_1_13 .pred_class_rf_res_1_17
   <fct>                    <fct>                    <fct>
 1 yes                      yes                      yes
 2 yes                      yes                      no
 3 no                       no                       no
 4 yes                      yes                      yes
 5 yes                      yes                      no
 6 no                       yes                      no
 7 yes                      yes                      yes
 8 yes                      no                       no
 9 no                       no                       yes
10 yes                      yes                      yes
   .pred_class_rf_res_1_19 .pred_class_rf_res_1_20 .pred_class_rf_res_1_21
   <fct>                    <fct>                    <fct>
 1 yes                      yes                      no
 2 yes                      yes                      no
 3 no                       no                       no
 4 yes                      yes                      yes
 5 no                       no                       yes
 6 no                       yes                      yes
 7 yes                      yes                      yes
 8 yes                      no                       no
 9 no                       no                       yes
10 yes                      yes                      yes
   .pred_class_rf_res_1_22 .pred_class_gb_res_1_6
   <fct>                    <fct>
 1 yes                      no
 2 no                       yes
 3 no                       no
 4 yes                      yes
 5 no                       no
 6 no                       no
 7 yes                      yes
 8 no                       yes
 9 no                       yes
10 yes                      yes
# i 47,213 more rows
```

```r
map(colnames(mimic_pred),
    ~mean(mimic_pred$los_long == pull(mimic_pred, .x))
    ) |>
  set_names(colnames(mimic_pred)) |>
  as_tibble() |>
  pivot_longer(c(everything(), -los_long))
```

```
# A tibble: 16 x 3
   los_long name                        value
      <dbl> <chr>                       <dbl>
 1        1 .pred_class                 0.591
 2        1 .pred_class_logit_res_1_001 0.561
 3        1 .pred_class_logit_res_1_101 0.561
 4        1 .pred_class_logit_res_1_201 0.561
 5        1 .pred_class_logit_res_1_301 0.561
 6        1 .pred_class_rf_res_1_06     0.581
 7        1 .pred_class_rf_res_1_07     0.584
 8        1 .pred_class_rf_res_1_08     0.584
 9        1 .pred_class_rf_res_1_09     0.587
10        1 .pred_class_rf_res_1_13     0.588
11        1 .pred_class_rf_res_1_17     0.587
12        1 .pred_class_rf_res_1_19     0.587
13        1 .pred_class_rf_res_1_20     0.587
14        1 .pred_class_rf_res_1_21     0.581
15        1 .pred_class_rf_res_1_22     0.584
16        1 .pred_class_gb_res_1_6      0.584
```

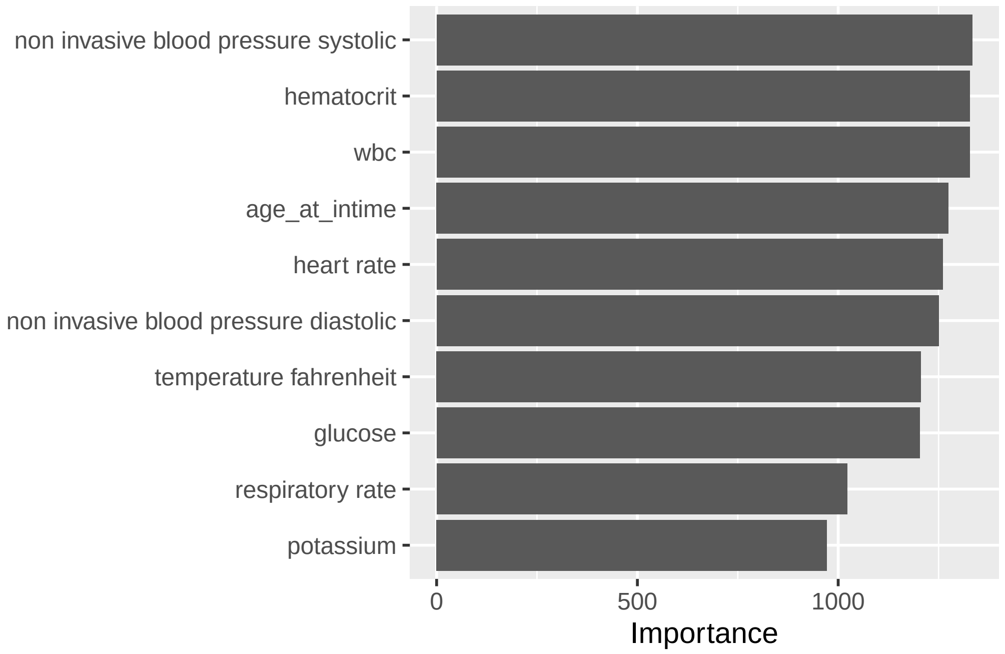
4. Compare model classification performance on the test set. Report both the area under ROC curve and accuracy for each machine learning algorithm and the model stacking. Interpret the results. What are the most important features in predicting long ICU stays? How do the models compare in terms of performance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

Solution:

```r
library(vip)
```

```
Attaching package: 'vip'


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

    vi
```

```r
best_rf <- rf_res |> select_best(metric = "roc_auc")

rf_fit <- rf_wf |>
  finalize_workflow(best_rf) |>
  fit(data = mimic_other)

rf_fit |>
  extract_fit_parsnip() |>
  vip(num_features = 10)
```



Logistic regression gives an AUC of 0.586;

SVM gives an AUC of 0.596;

xgboost gives the best performance with an AUC of 0.622.

model stacking(logistic + random forest + xgboost) gives an AUC of 0.591. boosting is the dominant model among all three.

xgboost provides the best result so far. The most important features predicting los_long according to rf(since it has the highest weight in stacking) are: hematocrit, wbc, glucose,blood pressures, age_at_intime and heart rate.

xgboost performed best in ROC-AUC, meaning it distinguishes long ICU stays most effectively.

Logistic regression provides the best interpretability since variable coefficients indicates the feature inpact directly.