hw5

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.

1.1 library

```r
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
Platform: x86_64-apple-darwin20 (64-bit)
Running under: macOS 15.3.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-
x86_64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-
x86_64/Resources/lib/libRlapack.dylib;  LAPACK version 3.11.0

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/Los_Angeles
tzcode source: internal

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

loaded via a namespace (and not attached):
 [1] htmlwidgets_1.6.4 compiler_4.3.3    fastmap_1.2.0     cli_3.6.3
 [5] tools_4.3.3       htmltools_0.5.8.1 rstudioapi_0.17.0 yaml_2.3.10
 [9] rmarkdown_2.28    knitr_1.49        jsonlite_1.8.9    xfun_0.48
[13] digest_0.6.37     rlang_1.1.4       evaluate_1.0.3
```

```r
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

```
library(ggplot2)
library(corrplot)
```

```
corrplot 0.95 loaded
```

```
library(lubridate)
library(miceRanger)
library(dplyr)
library(GGally)
```

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

```
library(gtsummary)
library(tidymodels)
```

```
── Attaching packages ────────────────────────────────────── tidymodels 1.2.0 ──
✔ broom        1.0.7     ✔ rsample      1.2.1
✔ dials        1.4.0     ✔ tune         1.2.1
✔ infer        1.0.7     ✔ workflows    1.1.4
✔ modeldata    1.4.0     ✔ workflowsets 1.1.0
✔ parsnip      1.3.0     ✔ yardstick    1.3.2
✔ recipes      1.1.1
── Conflicts ───────────────────────────────────────── tidymodels_conflicts() ──
✖ scales::discard() masks purrr::discard()
✖ dplyr::filter()   masks stats::filter()
✖ recipes::fixed()  masks stringr::fixed()
✖ dplyr::lag()      masks stats::lag()
✖ yardstick::spec() masks readr::spec()
✖ recipes::step()   masks stats::step()
• Learn how to get started at https://www.tidymodels.org/start/
```

```
library(yardstick)
library(dials)
library(kernlab)
```

```
Attaching package: 'kernlab'

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

    buffer

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
library(ggthemes)
library(naniar)
library(kableExtra)
```

```
Attaching package: 'kableExtra'

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

    group_rows
```

```r
library(stacks)
library(vip)
```

```
Attaching package: 'vip'

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

    vi
```

```r
library(h2o)
```

```
----------------------------------------------------------------------

Your next step is to start H2O:
    > h2o.init()

For H2O package documentation, ask for help:
    > ??h2o

After starting H2O, you can use the Web UI at http://localhost:54321
For more information visit https://docs.h2o.ai

----------------------------------------------------------------------



Attaching package: 'h2o'

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

    day, hour, month, week, year

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

    cor, sd, var

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

    &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
    colnames<-, ifelse, is.character, is.factor, is.numeric, log,
    log10, log1p, log2, round, signif, trunc
```

```
h2o.init(nthreads = -1, max_mem_size = "8G")
```

```
 Connection successful!

R is connected to the H2O cluster: 
    H2O cluster uptime:         6 hours 3 minutes 
    H2O cluster timezone:       America/Los_Angeles 
    H2O data parsing timezone:  UTC 
    H2O cluster version:        3.44.0.3 
    H2O cluster version age:    1 year, 2 months and 27 days 
    H2O cluster name:           H2O_started_from_R_guojiayi_rnw010 
    H2O cluster total nodes:    1 
    H2O cluster total memory:   6.76 GB 
    H2O cluster total cores:    8 
    H2O cluster allowed cores:  8 
    H2O cluster healthy:        TRUE 
    H2O Connection ip:          localhost 
    H2O Connection port:        54321 
    H2O Connection proxy:       NA 
    H2O Internal Security:      FALSE 
    R Version:                  R version 4.3.3 (2024-02-29) 

Warning in h2o.clusterInfo(): 
Your H2O cluster version is (1 year, 2 months and 27 days) old. There may be a newer
version available.
```

```
Please download and install the latest version from: https://h2o-
release.s3.amazonaws.com/h2o/latest_stable.html
```

1.2 Data preprocessing and feature engineering.

```r
icu_cohort <- readRDS("../hw4/mimiciv_shiny/mimiciv_icu_cohort.rds")
icu_cohort |>
  print(width = Inf)
```

```
# A tibble: 94,458 × 44
   subject_id  hadm_id stay_id.x
        <dbl>    <dbl>     <dbl>
 1   10000032 29079034  39553978
 2   10000690 25860671  37081114
 3   10000980 26913865  39765666
 4   10001217 24597018  37067082
 5   10001217 27703517  34592300
 6   10001725 25563031  31205490
 7   10001843 26133978  39698942
 8   10001884 26184834  37510196
 9   10002013 23581541  39060235
10   10002114 27793700  34672098
   first_careunit
   <fct>
 1 Medical Intensive Care Unit (MICU)
 2 Medical Intensive Care Unit (MICU)
 3 Medical Intensive Care Unit (MICU)
 4 Surgical Intensive Care Unit (SICU)
 5 Surgical Intensive Care Unit (SICU)
 6 Medical/Surgical Intensive Care Unit (MICU/SICU)
 7 Medical/Surgical Intensive Care Unit (MICU/SICU)
 8 Medical Intensive Care Unit (MICU)
 9 Cardiac Vascular Intensive Care Unit (CVICU)
10 Coronary Care Unit (CCU)
   last_careunit                                    intime
   <fct>                                            <dttm>
 1 Medical Intensive Care Unit (MICU)               2180-07-23 14:00:00
 2 Medical Intensive Care Unit (MICU)               2150-11-02 19:37:00
 3 Medical Intensive Care Unit (MICU)               2189-06-27 08:42:00
 4 Surgical Intensive Care Unit (SICU)              2157-11-20 19:18:02
 5 Surgical Intensive Care Unit (SICU)              2157-12-19 15:42:24
 6 Medical/Surgical Intensive Care Unit (MICU/SICU) 2110-04-11 15:52:22
 7 Medical/Surgical Intensive Care Unit (MICU/SICU) 2134-12-05 18:50:03
 8 Medical Intensive Care Unit (MICU)               2131-01-11 04:20:05
 9 Cardiac Vascular Intensive Care Unit (CVICU)     2160-05-18 10:00:53
10 Coronary Care Unit (CCU)                         2162-02-17 23:30:00
   outtime               los admittime           dischtime
   <dttm>              <dbl> <dttm>              <dttm>
 1 2180-07-23 23:50:47 0.410 2180-07-23 12:35:00 2180-07-25 17:55:00
 2 2150-11-06 17:03:17 3.89  2150-11-02 18:02:00 2150-11-12 13:45:00
 3 2189-06-27 20:38:27 0.498 2189-06-27 07:38:00 2189-07-03 03:00:00
 4 2157-11-21 22:08:00 1.12  2157-11-18 22:56:00 2157-11-25 18:00:00
 5 2157-12-20 14:27:41 0.948 2157-12-18 16:58:00 2157-12-24 14:55:00
 6 2110-04-12 23:59:56 1.34  2110-04-11 15:08:00 2110-04-14 15:00:00
 7 2134-12-06 14:38:26 0.825 2134-12-05 00:10:00 2134-12-06 12:54:00
 8 2131-01-20 08:27:30 9.17  2131-01-07 20:39:00 2131-01-20 05:15:00
 9 2160-05-19 17:33:33 1.31  2160-05-18 07:45:00 2160-05-23 13:30:00
10 2162-02-20 21:16:27 2.91  2162-02-17 22:32:00 2162-03-04 15:16:00
   deathtime           admission_type              admit_provider_id
   <dttm>              <fct>                       <chr>
 1 NA                  EW EMER.                    P060TX
 2 NA                  EW EMER.                    P26QQ4
 3 NA                  EW EMER.                    P060TX
 4 NA                  EW EMER.                    P3610N
 5 NA                  Other                       P276OU
 6 NA                  EW EMER.                    P32W56
 7 2134-12-06 12:54:00 URGENT                      P67ATB
 8 2131-01-20 05:15:00 OBSERVATION ADMIT           P49AFC
 9 NA                  SURGICAL SAME DAY ADMISSION P8286C
10 NA                  OBSERVATION ADMIT           P46834
   admission_location     discharge_location insurance language marital_status
   <fct>                  <fct>              <chr>     <chr>    <chr>
 1 EMERGENCY ROOM         HOME               Medicaid  English  WIDOWED
 2 EMERGENCY ROOM         Other              Medicare  English  WIDOWED
 3 EMERGENCY ROOM         HOME HEALTH CARE   Medicare  English  MARRIED
 4 EMERGENCY ROOM         HOME HEALTH CARE   Private   Other    MARRIED
 5 PHYSICIAN REFERRAL     HOME HEALTH CARE   Private   Other    MARRIED
 6 Other                  HOME               Private   English  MARRIED
 7 TRANSFER FROM HOSPITAL DIED               Medicare  English  SINGLE
 8 EMERGENCY ROOM         DIED               Medicare  English  MARRIED
 9 PHYSICIAN REFERRAL     HOME HEALTH CARE   Medicare  English  SINGLE
10 PHYSICIAN REFERRAL     HOME HEALTH CARE   Medicaid  English  <NA>
   edregtime           edouttime           hospital_expire_flag gender
   <dttm>              <dttm>                             <dbl> <chr>
 1 2180-07-23 05:54:00 2180-07-23 14:00:00                    0 F
 2 2150-11-02 11:41:00 2150-11-02 19:37:00                    0 F
 3 2189-06-27 06:25:00 2189-06-27 08:42:00                    0 F
 4 2157-11-18 17:38:00 2157-11-19 01:24:00                    0 F
 5 NA                  NA                                     0 F
 6 NA                  NA                                     0 F
 7 NA                  NA                                     1 M
 8 2131-01-07 13:36:00 2131-01-07 22:13:00                    1 F
 9 NA                  NA                                     0 F
10 2162-02-17 19:35:00 2162-02-17 23:30:00                    0 M
   anchor_age anchor_year anchor_year_group dod        stay_id.y Bicarbonate
        <dbl>       <dbl> <chr>             <date>         <dbl>       <dbl>
 1         52        2180 2014 - 2016       2180-09-09  39553978          25
 2         86        2150 2008 - 2010       2152-01-30  37081114          26
 3         73        2186 2008 - 2010       2193-08-26  39765666          21
 4         55        2157 2011 - 2013       NA          34592300          30
 5         55         2157 2011 - 2013       NA          34592300           30
 6         46         2110 2011 - 2013       NA          31205490           NA
 7         73         2131 2017 - 2019       2134-12-06  39698942           28
 8         68         2122 2008 - 2010       2131-01-20  37510196           30
 9         53         2156 2008 - 2010       NA          39060235           24
10         56         2162 2020 - 2022       2162-12-11  34672098           18
   Chloride Creatinine Glucose Potassium Sodium Hematocrit   wbc  stay_id    HR
      <dbl>      <dbl>   <dbl>     <dbl>  <dbl>      <dbl> <dbl>    <int> <dbl>
 1       95        0.7     102       6.7    126       41.1   6.9 39553978    91
 2      100        1        85       4.8    137       36.1   7.1 37081114    79
 3      109        2.3      89       3.9    144       27.3   5.3 39765666    77
 4      104        0.5      87       4.1    142       37.4   5.4 34592300    96
 5      104        0.5      87       4.1    142       37.4   5.4 34592300    96
 6       98       NA        NA       4.1    139       NA    NA   31205490    86
 7       97        1.3     131       3.9    138       31.4  10.4 39698942   118
 8       88        1.1     141       4.5    130       39.7  12.2 37510196    38
 9      102        0.9     288       3.5    137       34.9   7.2 39060235    80
10       NA        3.1      95       6.5    125       34.3  16.8 34672098   111
    NBPs  NBPd    RR    BT age_intime race  los_long
   <dbl> <dbl> <dbl> <dbl>      <dbl> <chr> <lgl>
 1    84    48    24  98.7         52 WHITE FALSE
 2   107    63    23  97.7         86 WHITE TRUE
 3   150    77    23  98           76 BLACK FALSE
 4   167    95    11  97.6         55 WHITE FALSE
 5   167    95    11  97.6         55 WHITE FALSE
 6    73    56    19  97.7         46 WHITE FALSE
 7   112    71    17  97.9         76 WHITE FALSE
 8   180    12    16  98.1         77 BLACK TRUE
 9   104    70    14  97.2         57 Other FALSE
10   112    80    20  97.9         56 Other TRUE
# ℹ 94,448 more rows
```

check the missing value

```
# check variables with more than 10000 missing values
icu_cohort %>%
  select_if(colSums(is.na(icu_cohort)) > 10000) %>%
  colnames()
```

```
[1] "deathtime" "edregtime" "edouttime" "dod"
```

```
# keep only variables with less than 10000 missing values
icu_cohort_discard <- icu_cohort %>%
  select_if(colSums(is.na(icu_cohort)) <= 10000)
```

delete data collected after ICU intime

```
icu_cohort_selected <- icu_cohort_discard |>
  select(-c("stay_id.y", "stay_id.x", "outtime", "dischtime",
```

```r
            "hospital_expire_flag", "last_careunit"))
print(icu_cohort_selected)
```

```
# A tibble: 94,458 × 34
   subject_id  hadm_id first_careunit                     intime                los
        <dbl>    <dbl> <fct>                              <dttm>              <dbl>
 1   10000032 29079034 Medical Intensive Care Unit (M… 2180-07-23 14:00:00 0.410
 2   10000690 25860671 Medical Intensive Care Unit (M… 2150-11-02 19:37:00 3.89
 3   10000980 26913865 Medical Intensive Care Unit (M… 2189-06-27 08:42:00 0.498
 4   10001217 24597018 Surgical Intensive Care Unit (… 2157-11-20 19:18:02 1.12
 5   10001217 27703517 Surgical Intensive Care Unit (… 2157-12-19 15:42:24 0.948
 6   10001725 25563031 Medical/Surgical Intensive Car… 2110-04-11 15:52:22 1.34
 7   10001843 26133978 Medical/Surgical Intensive Car… 2134-12-05 18:50:03 0.825
 8   10001884 26184834 Medical Intensive Care Unit (M… 2131-01-11 04:20:05 9.17
 9   10002013 23581541 Cardiac Vascular Intensive Car… 2160-05-18 10:00:53 1.31
10   10002114 27793700 Coronary Care Unit (CCU)        2162-02-17 23:30:00 2.91
# ℹ 94,448 more rows
# ℹ 29 more variables: admittime <dttm>, admission_type <fct>,
#   admit_provider_id <chr>, admission_location <fct>,
#   discharge_location <fct>, insurance <chr>, language <chr>,
#   marital_status <chr>, gender <chr>, anchor_age <dbl>, anchor_year <dbl>,
#   anchor_year_group <chr>, Bicarbonate <dbl>, Chloride <dbl>,
#   Creatinine <dbl>, Glucose <dbl>, Potassium <dbl>, Sodium <dbl>, …
```

figure out missing value

```r
# make a function to replace outliers to `NA`s
winsorize <- function(x, lower=0.01, upper=0.99) {
  qnt <- quantile(x, probs = c(lower, upper), na.rm = TRUE)
  x[x < qnt[1]] <- qnt[1]
  x[x > qnt[2]] <- qnt[2]
  x
}
# replace the extrem data with NA
icu_cohort_replace <- icu_cohort_selected %>%
  mutate(across(c("HR", "NBPs", "NBPd", "RR", "BT",
                "Bicarbonate", "Chloride", "Creatinine",
                "Glucose", "Potassium", "Sodium",
                "Hematocrit", "wbc"), winsorize))
# delete all rows with NA in los_long, marital_status
icu_cohort_replace <- icu_cohort_replace |>
  filter(!is.na(los_long)) |>
  filter(!is.na(marital_status)) |>
  # also delete labevent NA rows
  filter(!if_all(c("Bicarbonate", "Chloride", "Creatinine", "Glucose",
                 "Potassium", "Sodium", "Hematocrit", "wbc"), is.na))
# delete unnessary columns
icu_cohort_replace <- icu_cohort_replace |>
  select(-c("admit_provider_id", "discharge_location")) |>
  print()
```

```
# A tibble: 83,043 × 32
   subject_id  hadm_id first_careunit                     intime                los
        <dbl>    <dbl> <fct>                              <dttm>              <dbl>
 1   10000032 29079034 Medical Intensive Care Unit (M… 2180-07-23 14:00:00 0.410
 2   10000690 25860671 Medical Intensive Care Unit (M… 2150-11-02 19:37:00 3.89
 3   10000980 26913865 Medical Intensive Care Unit (M… 2189-06-27 08:42:00 0.498
 4   10001217 24597018 Surgical Intensive Care Unit (… 2157-11-20 19:18:02 1.12
 5   10001217 27703517 Surgical Intensive Care Unit (… 2157-12-19 15:42:24 0.948
 6   10001725 25563031 Medical/Surgical Intensive Car… 2110-04-11 15:52:22 1.34
 7   10001843 26133978 Medical/Surgical Intensive Car… 2134-12-05 18:50:03 0.825
 8   10001884 26184834 Medical Intensive Care Unit (M… 2131-01-11 04:20:05 9.17
 9   10002013 23581541 Cardiac Vascular Intensive Car… 2160-05-18 10:00:53 1.31
10   10002155 20345487 Medical Intensive Care Unit (M… 2131-03-09 21:33:00 0.859
# ℹ 83,033 more rows
# ℹ 27 more variables: admittime <dttm>, admission_type <fct>,
#   admission_location <fct>, insurance <chr>, language <chr>,
#   marital_status <chr>, gender <chr>, anchor_age <dbl>, anchor_year <dbl>,
#   anchor_year_group <chr>, Bicarbonate <dbl>, Chloride <dbl>,
#   Creatinine <dbl>, Glucose <dbl>, Potassium <dbl>, Sodium <dbl>,
#   Hematocrit <dbl>, wbc <dbl>, stay_id <int>, HR <dbl>, NBPs <dbl>, …
```

1.2.1 keep cleaned data in a file

```r
# fill in the NA value
if (file.exists("icu_cohort_filled.rds")){
  icu_cohort_filled <- read_rds("icu_cohort_filled.rds")
}else{
  imputed_data <- miceRanger(icu_cohort_replace,
                             m = 1,
                             max.depth = 8,
                             num.trees = 50)
  icu_cohort_filled <- completeData(imputed_data)
  icu_cohort_filled |>
    write_rds("icu_cohort_filled.rds")
}
icu_cohort_model <- as.data.frame(icu_cohort_filled) |>
  rename_with(~ gsub("^Dataset_1\\.", "", .x)) |>
  # Keep necessary columns
  select(
    "los_long",
    "gender", "marital_status", "race", "first_careunit",
    "Bicarbonate", "Chloride", "Creatinine", "Glucose",
    "Potassium", "Sodium", "Hematocrit", "wbc",
    "HR", "NBPs", "NBPd", "RR", "BT",
    "age_intime", "subject_id", "stay_id", "hadm_id"
  ) |>
  mutate(los_long = factor(los_long, levels = c(FALSE, TRUE))) |>
  mutate(age_intime = as.numeric(age_intime)) |>
  mutate(gender = as.factor(gender)) |>
  mutate(marital_status = as.factor(marital_status)) |>
  mutate(race = as.factor(race))
```

print the summary table

```
summary_table <- icu_cohort_model %>%
  select("los_long",
    "gender", "marital_status", "race", "first_careunit",
    "Bicarbonate", "Chloride", "Creatinine", "Glucose",
    "Potassium", "Sodium", "Hematocrit", "wbc",
    "HR", "NBPs", "NBPd", "RR", "BT") |>
  tbl_summary(by = los_long)
summary_table
```

Characteristic	**FALSE** N = 43,044[1]	**TRUE** N = 39,999[1]
gender		
F	19,403 (45%)	17,488 (44%)
M	23,641 (55%)	22,511 (56%)
marital_status		
DIVORCED	3,413 (7.9%)	3,228 (8.1%)
MARRIED	20,465 (48%)	19,495 (49%)
SINGLE	13,621 (32%)	12,177 (30%)
WIDOWED	5,545 (13%)	5,099 (13%)
race		
ASIAN	1,470 (3.4%)	1,318 (3.3%)
BLACK	5,314 (12%)	4,747 (12%)
HISPANIC	1,857 (4.3%)	1,600 (4.0%)
Other	3,696 (8.6%)	3,858 (9.6%)
WHITE	30,707 (71%)	28,476 (71%)
first_careunit		
Cardiac Vascular Intensive Care Unit (CVICU)	6,747 (16%)	6,539 (16%)
Coronary Care Unit (CCU)	4,535 (11%)	4,589 (11%)

[1] n (%); Median (Q1, Q3)

Characteristic	FALSE N = 43,044[1]	TRUE N = 39,999[1]
Medical Intensive Care Unit (MICU)	9,909 (23%)	8,483 (21%)
Medical/Surgical Intensive Care Unit (MICU/SICU)	8,089 (19%)	6,002 (15%)
Neuro Intermediate	1,895 (4.4%)	3,137 (7.8%)
Surgical Intensive Care Unit (SICU)	5,913 (14%)	5,674 (14%)
Trauma SICU (TSICU)	4,670 (11%)	4,058 (10%)
Other	1,286 (3.0%)	1,517 (3.8%)
Bicarbonate	24.0 (21.0, 27.0)	24.0 (21.0, 27.0)
Chloride	102 (98, 105)	102 (98, 105)
Creatinine	1.00 (0.80, 1.40)	1.00 (0.80, 1.60)
Glucose	118 (98, 154)	121 (100, 158)
Potassium	4.20 (3.90, 4.60)	4.20 (3.90, 4.70)
Sodium	139.0 (136.0, 141.0)	138.0 (135.0, 141.0)
Hematocrit	36 (30, 40)	34 (29, 40)
wbc	9.0 (6.6, 12.6)	9.4 (6.8, 13.4)
HR	85 (74, 99)	87 (75, 102)
NBPs	122 (106, 139)	120 (104, 138)
NBPd	68 (58, 80)	67 (56, 79)
RR	18 (15, 22)	19 (15, 23)
BT	98.10 (97.60, 98.60)	98.20 (97.60, 98.70)

[1] n (%); Median (Q1, Q3)

1.3 split the dataset

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.

```r
set.seed(203)
# arrange the data
icu_cohort_model_split <- icu_cohort_model |>
  arrange(subject_id, hadm_id, stay_id) |>
  select(-c("subject_id",
            "hadm_id",
            "stay_id"))
data_split <- initial_split(
  icu_cohort_model_split,
  strata = "los_long",
  prop = 0.5
)
data_split
```

```
<Training/Testing/Total>
<41521/41522/83043>
```

```r
icu_cohort_train <- training(data_split)
dim(icu_cohort_train)
```

```
[1] 41521    19
```

```r
icu_cohort_test <- testing(data_split)
dim(icu_cohort_test)
```

```
[1] 41522    19
```

```r
icu_cohort_train <- as.h2o(icu_cohort_train)
```

```

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```r
icu_cohort_test <- as.h2o(icu_cohort_test)
```

```

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

1.4 Logistic Regression

1.4.1 Model

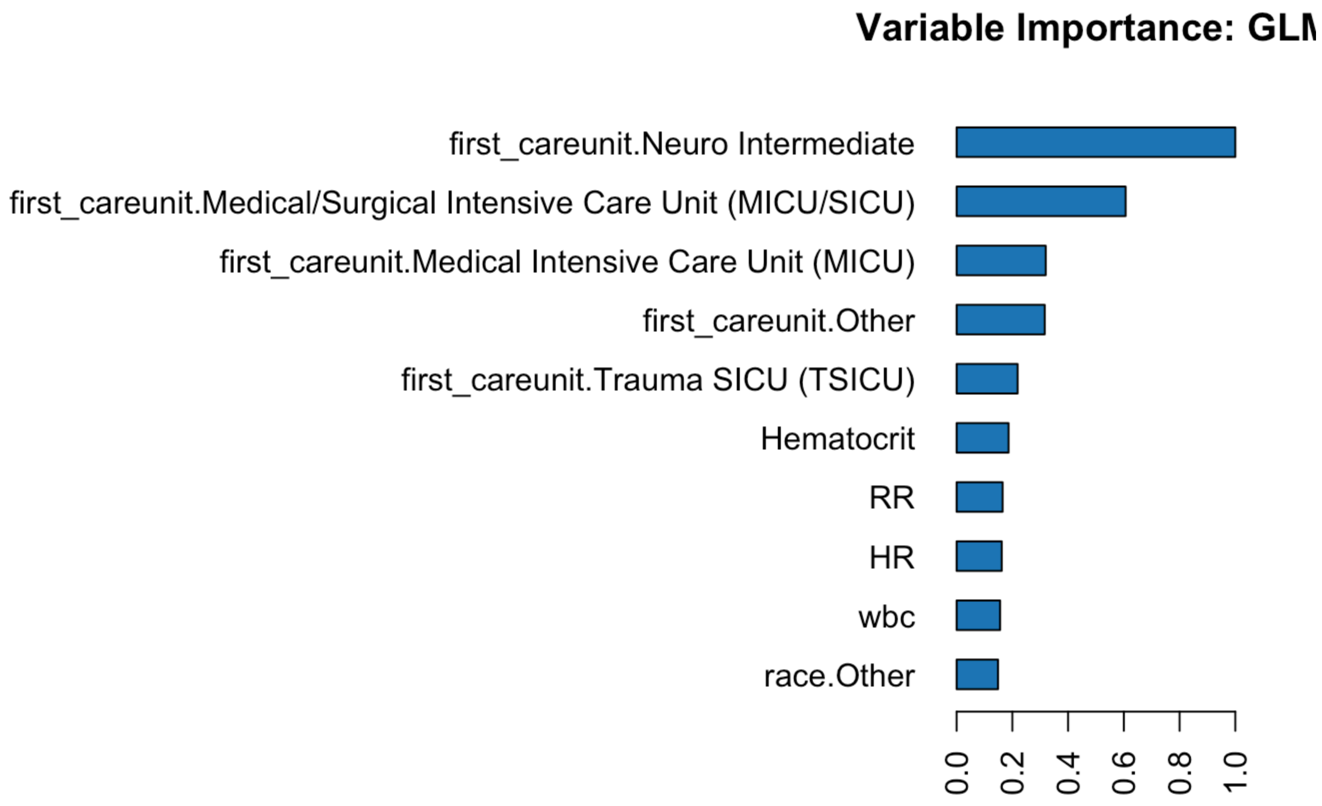
```r
logit_mod_h2o <- h2o.glm(
  x = c("gender", "marital_status", "race", "first_careunit",
        "Bicarbonate", "Chloride", "Creatinine", "Glucose",
        "Potassium", "Sodium", "Hematocrit", "wbc",
        "HR", "NBPs", "NBPd", "RR", "BT", "age_intime"),
  y = "los_long",
  training_frame = icu_cohort_train,
  family = "binomial",
  alpha = 0.5,
  lambda_search = TRUE,
  nfolds = 5,
  keep_cross_validation_predictions = TRUE
)
```

```
  |
  |                                                                      |   0%
  |
  |==============                                                        |  20%
  |
  |==========================                                            |  38%
  |
  |======================================================================| 100%
```

```
Warning in doTryCatch(return(expr), name, parentenv, handler): Reached maximum
number of iterations 47!
```

1.4.2 Visualize CV results:

```r
h2o.varimp_plot(logit_mod_h2o)
```



```r
h2o.performance(logit_mod_h2o)
```

```
H2OBinomialMetrics: glm
** Reported on training data. **

MSE:  0.2422779
RMSE:  0.4922174
LogLoss:  0.6774787
Mean Per-Class Error:  0.4933365
AUC:  0.5977081
AUCPR:  0.5695207
Gini:  0.1954162
R^2:  0.02958268
Residual Deviance:  56259.18
AIC:  56311.18

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
       FALSE  TRUE    Error           Rate
FALSE    473 21049 0.978022  =21049/21522
TRUE     173 19826 0.008650    =173/19999
Totals   646 40875 0.511115  =21222/41521

Maximum Metrics: Maximum metrics at their respective thresholds
```

```
                         metric threshold        value idx
1                        max f1  0.320247     0.651378 370
2                        max f2  0.243312     0.822930 397
3                  max f0point5  0.450966     0.560915 241
4                  max accuracy  0.484125     0.571614 198
5                 max precision  0.827308     1.000000   0
6                    max recall  0.243312     1.000000 397
7               max specificity  0.827308     1.000000   0
8              max absolute_mcc  0.473170     0.143657 212
9    max min_per_class_accuracy  0.476334     0.570579 208
10 max mean_per_class_accuracy  0.473170     0.571850 212
11                      max tns  0.827308 21522.000000   0
12                      max fns  0.827308 19998.000000   0
13                      max fps  0.225898 21522.000000 399
14                      max tps  0.243312 19999.000000 397
15                      max tnr  0.827308     1.000000   0
16                      max fnr  0.827308     0.999950   0
17                      max fpr  0.225898     1.000000 399
18                      max tpr  0.243312     1.000000 397

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
```

1.5 Random forest

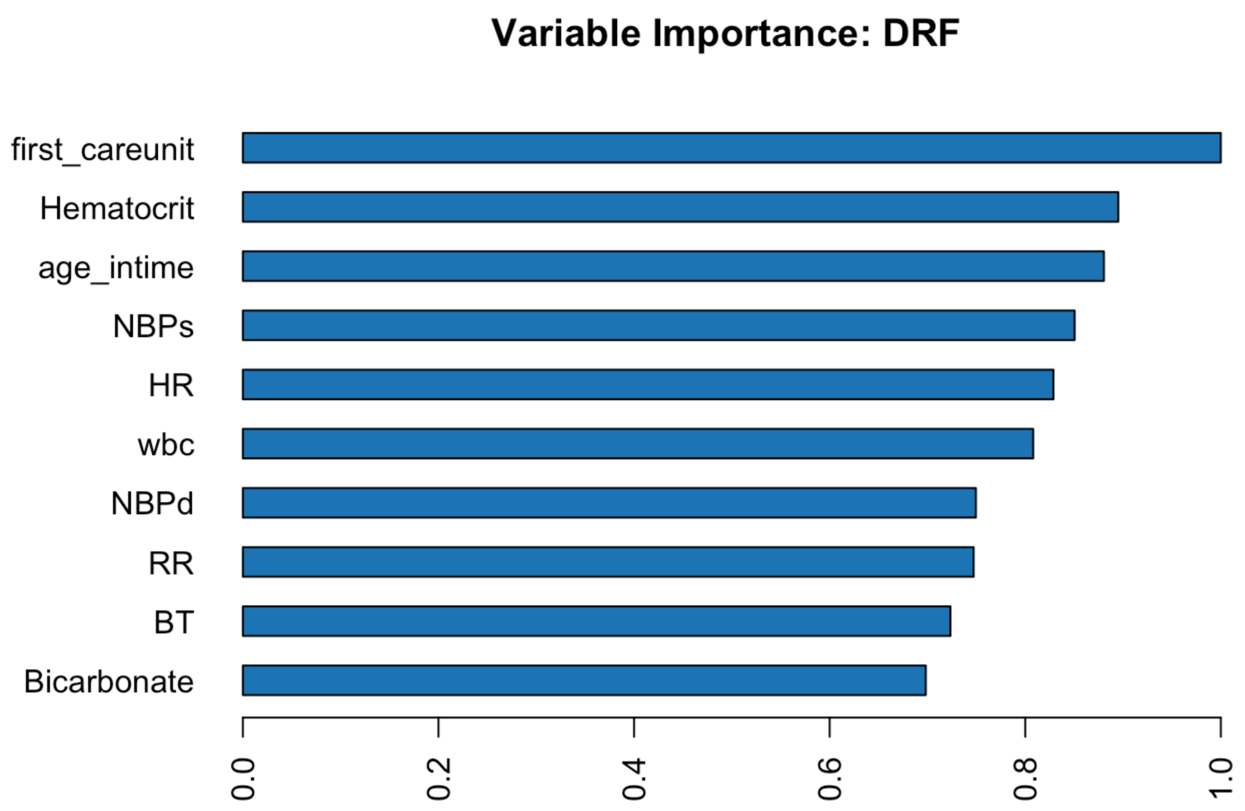
1.5.1 Model

```
rf_mod_h2o <- h2o.randomForest(
  x = c("gender", "marital_status", "race", "first_careunit",
        "Bicarbonate", "Chloride", "Creatinine", "Glucose",
        "Potassium", "Sodium", "Hematocrit", "wbc",
        "HR", "NBPs", "NBPd", "RR", "BT", "age_intime"),
  y = "los_long",
  training_frame = icu_cohort_train,
  ntrees = 100,
  mtries = -1,
  max_depth = 20,
  min_rows = 5,
  seed = 1234,
  nfolds = 5,
  balance_classes = TRUE,
  keep_cross_validation_predictions = TRUE
)
```

```
  |
  |                                                                      |   0%
  |
  |=                                                                     |   1%
  |
  |==                                                                    |   3%
  |
  |====                                                                  |   6%
  |
  |========                                                              |  11%
  |
  |==========                                                            |  15%
  |
  |============                                                          |  18%
  |
  |==============                                                        |  20%
  |
  |================                                                      |  23%
  |
  |===================                                                   |  28%
  |
  |======================                                                |  32%
  |
  |========================                                              |  35%
  |
  |==========================                                            |  37%
  |
  |============================                                          |  40%
  |
  |================================                                      |  45%
  |
  |===================================                                   |  50%
  |
  |====================================                                  |  52%
  |
  |======================================                                |  54%
  |
  |========================================                              |  57%
  |
  |===========================================                           |  62%
  |
  |=============================================                         |  65%
  |
  |===============================================                       |  67%
  |
  |=================================================                     |  70%
  |
  |==================================================                    |  72%
  |
  |======================================================                |  76%
  |
  |========================================================              |  80%
  |
  |==========================================================            |  83%
  |
  |============================================================          |  85%
  |
  |==============================================================        |  88%
  |
  |================================================================      |  92%
  |
  |===================================================================   |  96%
  |
  |======================================================================| 100%
```

1.5.2 Visualize CV results

```
h2o.varimp_plot(rf_mod_h2o)
```



1.6 XGBoost

1.6.1 Model

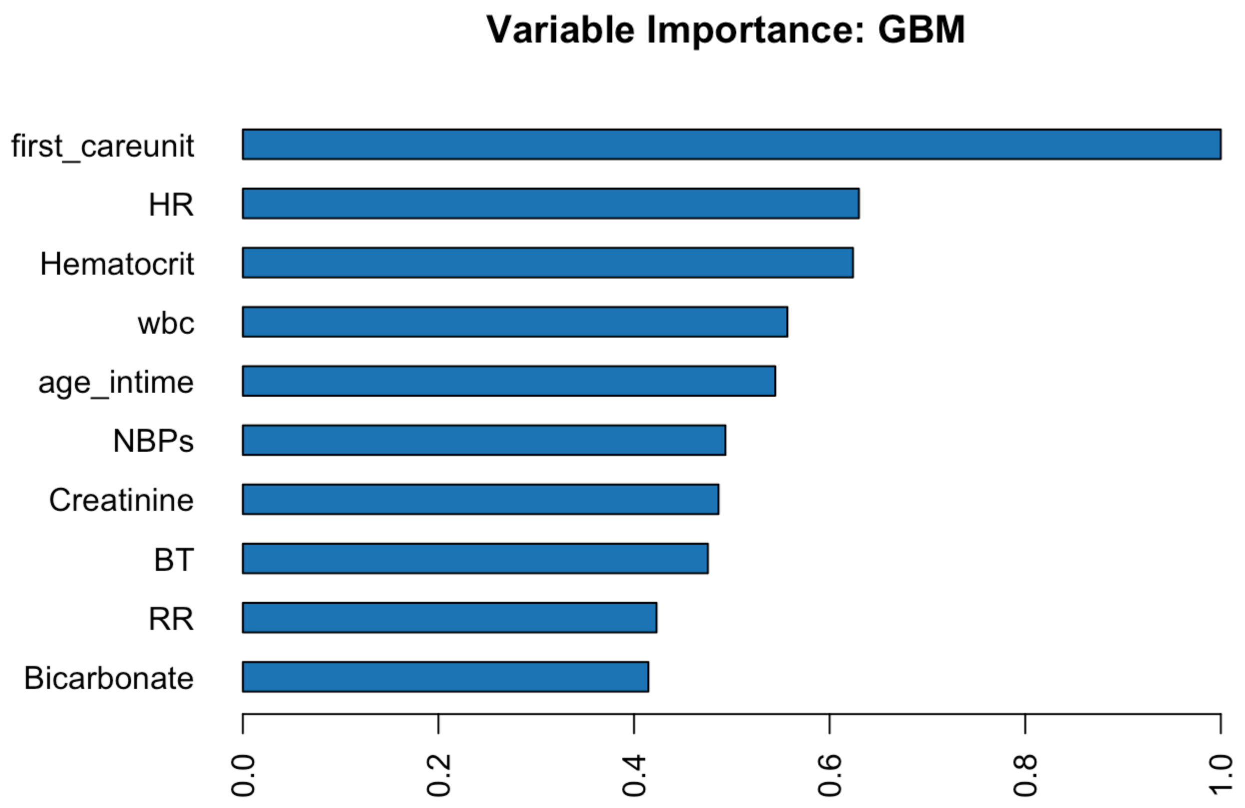
```
gb_mod_h2o <- h2o.gbm(
  x = c("gender", "marital_status", "race", "first_careunit",
```

```r
        "Bicarbonate", "Chloride", "Creatinine", "Glucose",
        "Potassium", "Sodium", "Hematocrit", "wbc",
        "HR", "NBPs", "NBPd", "RR", "BT", "age_intime"),
  y = "los_long",
  training_frame = icu_cohort_train,
  ntrees = 200,
  max_depth = 6,
  learn_rate = 0.05,
  sample_rate = 0.8,
  col_sample_rate = 0.8,
  seed = 1234,
  nfolds = 5,
  keep_cross_validation_predictions = TRUE
)
```

```
  |                                                                            
  |                                                                      |   0%
  |                                                                            
  |=                                                                     |   2%
  |                                                                            
  |====                                                                  |   5%
  |                                                                            
  |=========                                                             |  13%
  |                                                                            
  |=================                                                     |  24%
  |                                                                            
  |=======================                                               |  33%
  |                                                                            
  |==========================                                            |  37%
  |                                                                            
  |=============================                                         |  41%
  |                                                                            
  |====================================                                  |  52%
  |                                                                            
  |============================================                          |  63%
  |                                                                            
  |===============================================                       |  68%
  |                                                                            
  |=================================================                     |  70%
  |                                                                            
  |====================================================                  |  75%
  |                                                                            
  |==========================================================            |  83%
  |                                                                            
  |=============================================================         |  87%
  |                                                                            
  |================================================================      |  91%
  |                                                                            
  |===================================================================== |  99%
  |
  |======================================================================| 100%
```

1.6.2 Visualize CV results

```
h2o.varimp_plot(gb_mod_h2o)
```



1.7 Model Stacking

```
feature_columns <- setdiff(names(icu_cohort_train), "los_long")

h2o_stacked_model <- h2o.stackedEnsemble(
  x = feature_columns,
  y = "los_long",
  training_frame = icu_cohort_train,
  base_models = list(rf_mod_h2o, gb_mod_h2o, logit_mod_h2o)
)
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

1.7.1 Predict

```r
h2o_stacked_predictions <- h2o.predict(h2o_stacked_model, icu_cohort_test) |>
  print()
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
  predict     FALSE      TRUE
1    TRUE 0.5125255 0.4874745
2    TRUE 0.6031984 0.3968016
3   FALSE 0.6697046 0.3302954
4   FALSE 0.6697046 0.3302954
5    TRUE 0.5583365 0.4416635
6    TRUE 0.5873285 0.4126715

[41522 rows x 3 columns]
```

1.7.2 Model Performance

```r
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following object is masked from 'package:h2o':

    var

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

    cov, smooth, var
```

```r
h2o.performance(h2o_stacked_model, newdata = icu_cohort_test)
```

```
H2OBinomialMetrics: stackedensemble

MSE:  0.2352445
RMSE:  0.4850201
LogLoss:  0.6627929
Mean Per-Class Error:  0.463774
```

```
AUC:  0.6369982
AUCPR:  0.6102181
Gini:  0.2739963

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
       FALSE  TRUE    Error          Rate
FALSE   2573 18949 0.880448  =18949/21522
TRUE     942 19058 0.047100    =942/20000
Totals  3515 38007 0.479047  =19891/41522

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold        value idx
1                       max f1  0.335106     0.657093 338
2                       max f2  0.192474     0.822930 398
3                 max f0point5  0.454596     0.585274 229
4                 max accuracy  0.489978     0.598068 196
5                max precision  0.834882     1.000000   0
6                   max recall  0.192474     1.000000 398
7              max specificity  0.834882     1.000000   0
8             max absolute_mcc  0.454596     0.196925 229
9   max min_per_class_accuracy  0.474861     0.596550 210
10 max mean_per_class_accuracy  0.454596     0.597682 229
11                     max tns  0.834882 21522.000000   0
12                     max fns  0.834882 19996.000000   0
13                     max fps  0.187252 21522.000000 399
14                     max tps  0.192474 20000.000000 398
15                     max tnr  0.834882     1.000000   0
16                     max fnr  0.834882     0.999800   0
17                     max fpr  0.187252     1.000000 399
18                     max tpr  0.192474     1.000000 398

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
```

```r
# stacking model
perf_stack <- h2o.performance(h2o_stacked_model, newdata = icu_cohort_test)
auc_stack <- h2o.auc(perf_stack)

# rf model
perf_rf <- h2o.performance(rf_mod_h2o, newdata = icu_cohort_test)
auc_rf <- h2o.auc(perf_rf)

# gb model
perf_gb <- h2o.performance(gb_mod_h2o, newdata = icu_cohort_test)
auc_gb <- h2o.auc(perf_gb)

# logit model
perf_logit <- h2o.performance(logit_mod_h2o, newdata = icu_cohort_test)
auc_logit <- h2o.auc(perf_logit)
```

```
# table
auc_table <- data.frame(
  Model = c("Stacked Ensemble", "Random Forest",
            "GBM", "Logistic Regression"),
  AUC = c(auc_stack, auc_rf, auc_gb, auc_logit)
) |>
  print()
```

```
                Model       AUC
1    Stacked Ensemble 0.6369982
2       Random Forest 0.6263728
3                 GBM 0.6341979
4 Logistic Regression 0.6030615
```

conclusion The table above compares the AUC (Area Under the ROC Curve) values for four different classification models—Stacked Ensemble, Random Forest, GBM (Gradient Boosting Machine), and Logistic Regression. The Stacked Ensemble achieves the highest AUC at 0.6370, followed by GBM at 0.6342, Random Forest at 0.6264, and Logistic Regression at 0.6031.The stacked model slightly outperforms the individual algorithms.

1.7.3 Predicting compare

```
# Logistic Regression
perf_logit <- h2o.performance(logit_mod_h2o, newdata = icu_cohort_test)
cm_logit <- h2o.confusionMatrix(perf_logit, thresholds = 0.5)
```

```
Warning in h2o.find_row_by_threshold(object, t): Could not find exact
threshold: 0.5 for this set of metrics; using closest threshold found:
0.500178129651075. Run `h2o.predict` and apply your desired threshold on a
probability column.
```

```
cat("\n--- Logistic Regression Confusion Matrix (threshold=0.5) ---\n")
```

```

--- Logistic Regression Confusion Matrix (threshold=0.5) ---
```

```
print(cm_logit)
```

```
Confusion Matrix (vertical: actual; across: predicted)  @ threshold = 0.500178129651075:
       FALSE  TRUE    Error          Rate
FALSE  14770  6752 0.313725   =6752/21522
TRUE   10891  9109 0.544550  =10891/20000
Totals 25661 15861 0.424907  =17643/41522
```

```
# Random Forest
perf_rf <- h2o.performance(rf_mod_h2o, newdata = icu_cohort_test)
cm_rf <- h2o.confusionMatrix(perf_rf, thresholds = 0.5)
```

```
Warning in h2o.find_row_by_threshold(object, t): Could not find exact
threshold: 0.5 for this set of metrics; using closest threshold found:
0.499992401930058. Run `h2o.predict` and apply your desired threshold on a
probability column.
```

```
cat("\n--- Random Forest Confusion Matrix (threshold=0.5) ---\n")
```

```

--- Random Forest Confusion Matrix (threshold=0.5) ---
```

```
print(cm_rf)
```

```
Confusion Matrix (vertical: actual; across: predicted)  @ threshold = 0.499992401930058:
       FALSE  TRUE    Error          Rate
FALSE  14726  6796 0.315770   =6796/21522
TRUE   10198  9802 0.509900  =10198/20000
Totals 24924 16598 0.409277  =16994/41522
```

```
# GBM
perf_gb <- h2o.performance(gb_mod_h2o, newdata = icu_cohort_test)
cm_gb <- h2o.confusionMatrix(perf_gb, thresholds = 0.5)
```

```
Warning in h2o.find_row_by_threshold(object, t): Could not find exact
threshold: 0.5 for this set of metrics; using closest threshold found:
0.500408734793412. Run `h2o.predict` and apply your desired threshold on a
probability column.
```

```
cat("\n--- GBM Confusion Matrix (threshold=0.5) ---\n")
```

```

--- GBM Confusion Matrix (threshold=0.5) ---
```

```
print(cm_gb)
```

```
Confusion Matrix (vertical: actual; across: predicted)  @ threshold = 0.500408734793412:
       FALSE  TRUE    Error          Rate
FALSE  14496  7026 0.326457   =7026/21522
TRUE    9744 10256 0.487200   =9744/20000
Totals 24240 17282 0.403882  =16770/41522
```

```
# Stacked Ensemble
perf_stack <- h2o.performance(h2o_stacked_model, newdata = icu_cohort_test)
cm_stack <- h2o.confusionMatrix(perf_stack, thresholds = 0.5)
```

```
Warning in h2o.find_row_by_threshold(object, t): Could not find exact
threshold: 0.5 for this set of metrics; using closest threshold found:
```

```
0.499652532287864. Run `h2o.predict` and apply your desired threshold on a
probability column.
```

```
cat("\n--- Stacked Ensemble Confusion Matrix (threshold=0.5) ---\n")
```

```

--- Stacked Ensemble Confusion Matrix (threshold=0.5) ---
```

```
print(cm_stack)
```

```
Confusion Matrix (vertical: actual; across: predicted)  @ threshold = 0.499652532287864:
       FALSE  TRUE    Error          Rate
FALSE  14551  6971 0.323901   =6971/21522
TRUE    9751 10249 0.487550   =9751/20000
Totals 24302 17220 0.402726  =16722/41522
```

conclusion The Stacked Ensemble achieves the lowest overall error and the highest AUC, indicating that combining multiple base learners yields better predictive performance. Random Forest and GBM both outperform Logistic Regression, reflecting the benefit of more flexible, non-linear modeling. However, each model still shows substantial difficulty correctly identifying positive cases, as evidenced by higher error rates in the TRUE class. Overall, these results suggest that ensemble methods—especially stacking—can provide incremental yet meaningful improvements over individual algorithms.

1.7.4 ROC plot

```
true_labels <- as.vector(icu_cohort_test$los_long)
get_roc_df <- function(model, test_frame, true_labels, pos_class = "TRUE") {
  preds <- h2o.predict(model, test_frame)
  pred_prob <- as.vector(preds[[pos_class]])
  # use pROC
  roc_obj <- roc(true_labels, pred_prob)
  roc_df <- data.frame(
    fpr = 1 - rev(roc_obj$specificities),
    tpr = rev(roc_obj$sensitivities)
  )
  return(roc_df)
}

# get ROC for each model
roc_logit <- get_roc_df(logit_mod_h2o, icu_cohort_test, true_labels)
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

Setting levels: control = FALSE, case = TRUE
Setting direction: controls < cases
```

```r
roc_rf    <- get_roc_df(rf_mod_h2o, icu_cohort_test, true_labels)
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

Setting levels: control = FALSE, case = TRUE
Setting direction: controls < cases
```

```r
roc_gb    <- get_roc_df(gb_mod_h2o, icu_cohort_test, true_labels)
```

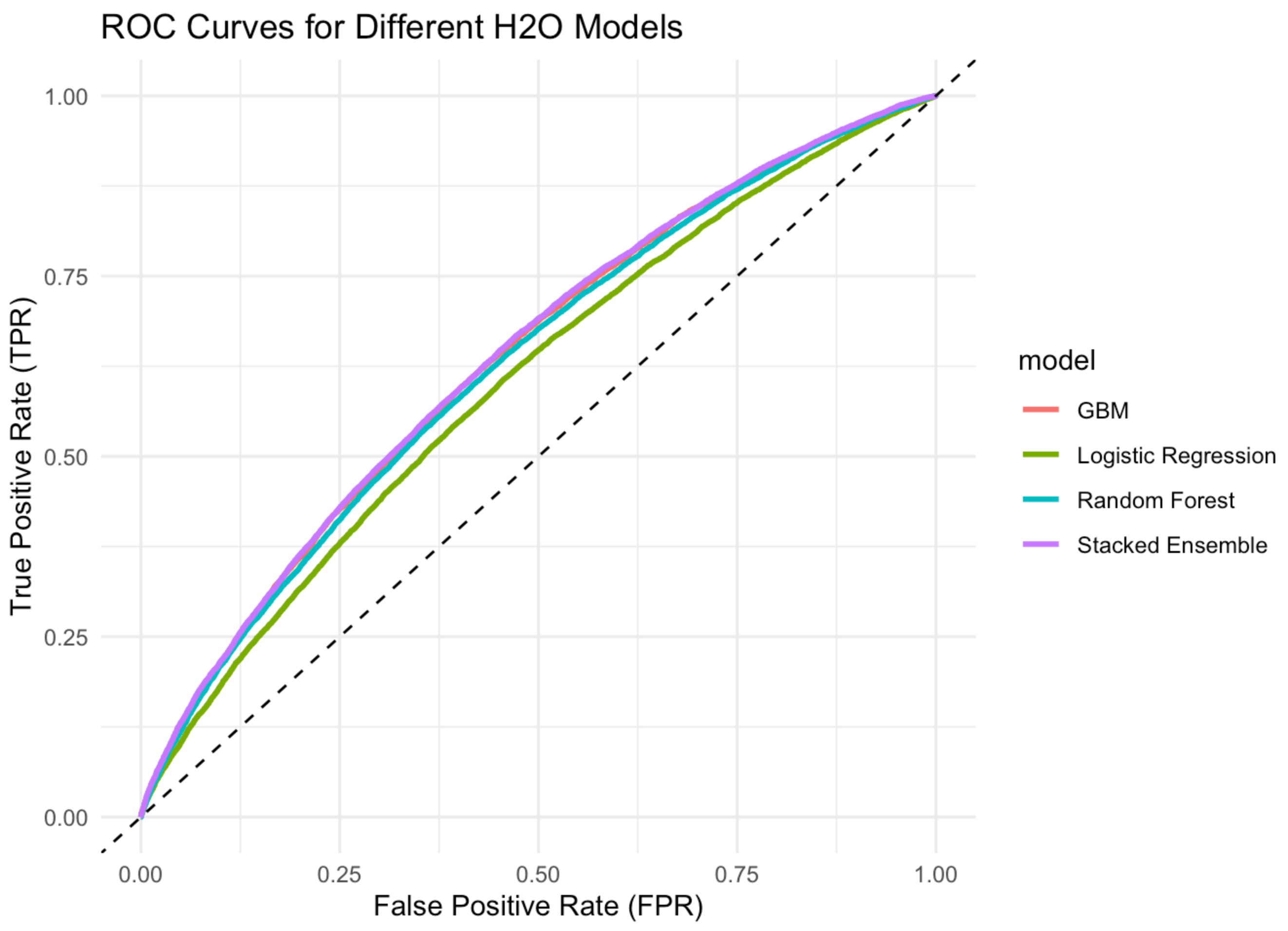
```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

Setting levels: control = FALSE, case = TRUE
Setting direction: controls < cases
```

```r
roc_stack <- get_roc_df(h2o_stacked_model, icu_cohort_test, true_labels)
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

Setting levels: control = FALSE, case = TRUE
Setting direction: controls < cases
```

```r
roc_logit$model <- "Logistic Regression"
roc_rf$model    <- "Random Forest"
roc_gb$model    <- "GBM"
roc_stack$model <- "Stacked Ensemble"

# merge all ROC data
roc_data <- rbind(roc_logit, roc_rf, roc_gb, roc_stack)

# ggplot
ggplot(roc_data, aes(x = fpr, y = tpr, color = model)) +
  geom_line(size = 1) +
  geom_abline(linetype = "dashed", color = "black") +
  labs(title = "ROC Curves for Different H2O Models",
       x = "False Positive Rate (FPR)",
       y = "True Positive Rate (TPR)") +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
ℹ Please use `linewidth` instead.
```



conclusion From the figure it can be seen that stacking model has the best ROC curve performance, which is similar to that of GBM, random forests are slightly inferior to them, and logistic regression has the worst ROC performance.

What are the most important features in predicting long ICU stays? How do the models compare in terms of performance and interpretability? From the feature importance analysis, we can see all three models agree that first careunit is one of the most important features in predicting long ICU stays.The last lab measurements before the ICU stay and first vital measurements during ICU stay are also in top important features. This suggests that the last lab measurements before the ICU stay and first vital measurements during ICU stay are important indicators of the patient's condition and may be useful in predicting long ICU stays. In this case, the trade-off between performance and interpretability is clear. While the model stacking approach gives the best ROC AUC, it does so at the cost of interpretability. On the other hand, logistic regression offers ease of interpretation but doesn't perform as well. The gradient boosting model presents a good balance, with relatively high performance metrics and a degree of interpretability through feature importance scores.