project model 1

Jingjing Li

2022-12-09

Helper packages

```
library(ROCR)
library(ggplot2)
library(lattice)
library(caret)
library(modeldata)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rsample)
library(recipes)
```

```
##
## Attaching package: 'recipes'

## The following object is masked from 'package:stats':
##
##     step
```

```
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:caret':
##
##     lift
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8     v stringr 1.5.0
## v tidyr   1.2.1     v forcats 0.5.2
## v readr   2.1.3
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x stringr::fixed() masks recipes::fixed()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(rpart)
library(rpart.plot)
library(readr)
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

```
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 object is masked from 'package:pROC':
##
##     var
##
## 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
```

process the data

```
df = read.csv("radiomics_completedata.csv")
dim(df)
```

```
## [1] 197 431
```

```
#check for null or missing value
any(is.na(df))
```

```
## [1] FALSE
```

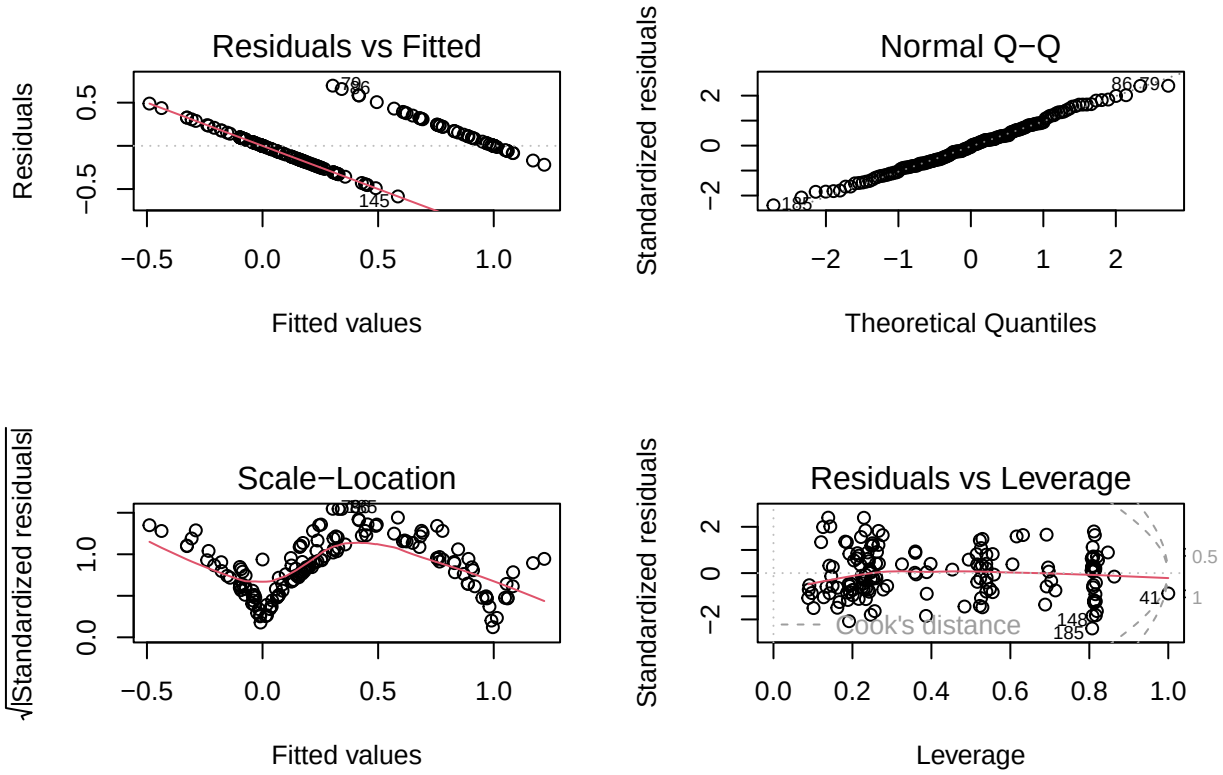
```
#check for normality,

m1<-lm(df$Failure.binary~., data=df)
par(mfrow=c(2,2))
plot(m1)#look at the QQplot,it's normal distribution.
```

```
## Warning: not plotting observations with leverage one:
##   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```




```
#correlation of the whole data, use the numerical data set df1

df1<-select(df,-c(Institution, Failure.binary))
cor_df1<-cor(df1)

############omit missing (if there is any)##############
#####convert binary column to factor for train model#################
df<-na.omit(df)
df$Failure.binary<-as.factor(df$Failure.binary)
```

train te model

Split the data into training (80) and testing (20). Make sure we have consistent categorical levels.

```
#split data
split = initial_split(df,prop = 0.8 ,strata = "Failure.binary")

ames_train <- training(split)
ames_test <- testing(split)

# Make sure we have consistent categorical levels on training data
blueprint <- recipe(Failure.binary ~ ., data = ames_train) %>%
  step_other(all_nominal(), threshold = 0.005)
```

```
#Make sure we have consistent categorical levels on test data
blueprint <- recipe(Failure.binary ~ ., data = ames_test) %>%
  step_other(all_nominal(), threshold = 0.005)
```

Convert the training & test sets to an h2o object

```
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         2 days 6 hours
##     H2O cluster timezone:       America/Toronto
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.38.0.1
##     H2O cluster version age:    2 months and 20 days
##     H2O cluster name:           H2O_started_from_R_ljingj_gwf961
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.50 GB
##     H2O cluster total cores:    6
##     H2O cluster allowed cores:  6
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.2.2 (2022-10-31 ucrt)
```

```
train_h2o <- prep(blueprint, training = ames_train, retain = TRUE) %>%
  juice() %>%
  as.h2o()
```

```
##   |                                                                      |
```

```
test_h2o <- prep(blueprint, training = ames_train) %>%
  bake(new_data = ames_test) %>%
  as.h2o()
```

```
##   |                                                                      |
```

Get response and feature names

```
Y <- "Failure.binary"
X <- setdiff(names(ames_train), Y)
```

Train & cross-validate a GLM model

```
best_glm <- h2o.glm(
  x = X, y = Y, training_frame = train_h2o, alpha = 0.1,
  remove_collinear_columns = TRUE, nfolds = 10, fold_assignment = "Modulo",
  family= c("binomial"), keep_cross_validation_predictions = TRUE, seed = 123
)
```

```
##   |                                                                      |
```

Train & cross-validate a RF model

```
#I had adjust ntrees to a very small number, otherwise my computer crashes.
best_rf <- h2o.randomForest(
  x = X, y = Y, training_frame = train_h2o, ntrees = 500, mtries = 20,
  max_depth = 30, min_rows = 1, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123, stopping_rounds = 50, score_each_iteration = T,
  stopping_tolerance = 0
)
```

```
##   |                                                                      |
```

Train & cross-validate a GBM model

```
#I had adjust ntrees to a very small number, otherwise my computer crashes
best_gbm <- h2o.gbm(
  x = X, y = Y, training_frame = train_h2o, ntrees = 500, learn_rate = 0.01,
  max_depth = 7, min_rows = 5, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123, stopping_rounds = 50, score_each_iteration = T,
  stopping_tolerance = 0
)
```

```
##   |                                                                      |
```

```
base_models <- list(best_gbm,best_glm,best_rf)
ensemble <- h2o.stackedEnsemble(x = X,
                                y = Y,
                                training_frame = train_h2o,
                                base_models = base_models)
```

```
##   |                                                                      |
```

Print the AUC values during training

```
get_auc_train <- function(model) {
  results <- h2o.performance(model, newdata = train_h2o)
  results@metrics$AUC
}
list(ensemble) %>%
  purrr::map_dbl(get_auc_train)
```

```
## [1] 1
```

Eval ensemble performance on a test set

```
# Eval ensemble performance on a test set
h2o.performance(ensemble, newdata = test_h2o)
```

```
## H2OBinomialMetrics: stackedensemble
##
## MSE:  0.06878508
## RMSE:  0.2622691
## LogLoss:  0.2427436
## Mean Per-Class Error:  0.05769231
## AUC:  0.9725275
## AUCPR:  0.9483235
## Gini:  0.9450549
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##         0  1    Error   Rate
## 0      23  3 0.115385  =3/26
## 1       0 14 0.000000  =0/14
## Totals 23 17 0.075000  =3/40
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold     value idx
## 1                       max f1  0.256738  0.903226  16
## 2                       max f2  0.256738  0.958904  16
## 3                 max f0point5  0.939669  0.900000   8
## 4                 max accuracy  0.513251  0.925000  14
## 5                max precision  0.957703  1.000000   0
## 6                   max recall  0.256738  1.000000  16
## 7              max specificity  0.957703  1.000000   0
## 8             max absolute_mcc  0.256738  0.853526  16
## 9   max min_per_class_accuracy  0.513251  0.923077  14
## 10 max mean_per_class_accuracy  0.256738  0.942308  16
## 11                     max tns  0.957703 26.000000   0
## 12                     max fns  0.957703 13.000000   0
## 13                     max fps  0.050223 26.000000  39
## 14                     max tps  0.256738 14.000000  16
## 15                     max tnr  0.957703  1.000000   0
## 16                     max fnr  0.957703  0.928571   0
## 17                     max fpr  0.050223  1.000000  39
## 18                     max tpr  0.256738  1.000000  16
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

Print the AUC value during testing

```
get_auc_test <- function(model) {
  results <- h2o.performance(model, newdata = test_h2o)
  results@metrics$AUC
}
list(ensemble) %>%
  purrr::map_dbl(get_auc_test)
```

```
## [1] 0.9725275
```

Print top 20 important feature during training

```
#########cannot print, also tried varImp(), summary.gbm(),
vip(ensemble,num_features=20)
```

```
## Warning: This model doesn't have variable importances
```

```
## Error in `tib[1L:2L]`:
## ! Can't subset columns past the end.
## i Locations 1 and 2 don't exist.
## i There are only 0 columns.
```