MODEL 1 - Random Forest

FENDAWN F. RECENTES

12/15/2022

Helper and Modeling Packages

```
library(dplyr)    # for data wrangling
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
##
## 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(ggplot2)  # for awesome graphics
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(ranger)   # a c++ implementation of random forest
```

```
## Warning: package 'ranger' was built under R version 4.1.3
```

```
library(h2o)      # a java-based implementation of random forest
```

```
## Warning: package 'h2o' was built under R version 4.1.3
```

```
##
## ----------------------------------------------------------------------
##
## 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: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
```

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.1.3
```

```
h2o.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     C:\Users\MSU-TC~1\AppData\Local\Temp\RtmpKEkCjA\file1e803dde3dda/h2o_MSU_TCTO_OVCAA_started_from_
##     C:\Users\MSU-TC~1\AppData\Local\Temp\RtmpKEkCjA\file1e801346244b/h2o_MSU_TCTO_OVCAA_started_from_
##
##
## Starting H2O JVM and connecting:  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 seconds 331 milliseconds
##     H2O cluster timezone:       Asia/Manila
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.38.0.1
##     H2O cluster version age:    2 months and 27 days
##     H2O cluster name:           H2O_started_from_R_MSU-TCTO_OVCAA_mvc880
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.93 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 4.1.2 (2021-11-01)
```

Load and view radiomics data set

```
radiomics <- read_csv("C:\\Users\\MSU-TCTO OVCAA\\Documents\\normalRad.csv")
```

```
## Rows: 197 Columns: 431
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): Institution
## dbl (430): Failure.binary, Failure, Entropy_cooc.W.ADC, GLNU_align.H.PET, Mi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

DATA PREPARATION AND SPLITTING

Split the data intro training (80%) and testing (20%) stratified in Failure.binary column

```
set.seed(123)
split = initial_split(radiomics,prop = 0.8 ,strata = "Failure.binary")
radiomics_train <- training(split)
radiomics_test  <- testing(split)
```

Convert target variable to a factor form

```
radiomics$Failure.binary=as.factor(radiomics$Failure.binary)
```

Number of features

```
n_features <- length(setdiff(names(radiomics_train), "Failure.binary"))
```

Train a default random forest model

```
radiomics_rf1 <- ranger(
  Failure.binary ~ .,
  data = radiomics_train,
  mtry = floor(n_features / 3),
  respect.unordered.factors = "order",
  seed = 123
)
```

Get OOB RMSE

```
(default_rmse <- sqrt(radiomics_rf1$prediction.error))
```

```
## [1] 0.2983203
```

Create hyperparameter grid

```
hyper_grid <- expand.grid(
  mtry = floor(n_features * c(.05, .15, .25, .333, .4)),
  min.node.size = c(1, 3, 5, 10),
  replace = c(TRUE, FALSE),
  sample.fraction = c(.5, .63, .8),
  rmse = NA
)
```

Execute full cartesian grid search

```
# execute full cartesian grid search
for(i in seq_len(nrow(hyper_grid))) {
  # fit model for ith hyperparameter combination
  fit <- ranger(
    formula         = Failure.binary ~ .,
    data            = radiomics_train,
    num.trees       = n_features * 10,
    mtry            = hyper_grid$mtry[i],
    min.node.size   = hyper_grid$min.node.size[i],
    replace         = hyper_grid$replace[i],
    sample.fraction = hyper_grid$sample.fraction[i],
    verbose         = FALSE,
    seed            = 123,
    respect.unordered.factors = 'order',
  )
  # export OOB error
  hyper_grid$rmse[i] <- sqrt(fit$prediction.error)
}
```

Assess top 10 models

```
hyper_grid %>%
  arrange(rmse) %>%
  mutate(perc_gain = (default_rmse - rmse) / default_rmse * 100) %>%
  head(10)
```

```
##    mtry min.node.size replace sample.fraction      rmse perc_gain
## 1   172            10   FALSE            0.80 0.2935338 1.6044890
## 2   172             5   FALSE            0.80 0.2940303 1.4380506
## 3   172             3   FALSE            0.80 0.2943725 1.3233447
## 4   172             1   FALSE            0.80 0.2944050 1.3124522
## 5   172             5   FALSE            0.63 0.2958944 0.8131878
## 6   172            10   FALSE            0.63 0.2961481 0.7281476
## 7   172             1   FALSE            0.63 0.2962982 0.6778218
## 8   172             3   FALSE            0.63 0.2963064 0.6750887
## 9   143            10   FALSE            0.80 0.2969303 0.4659584
## 10  143             3   FALSE            0.80 0.2973159 0.3366927
```

```
h2o.no_progress()
h2o.init(max_mem_size = "5g")
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 minutes 34 seconds
##     H2O cluster timezone:       Asia/Manila
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.38.0.1
##     H2O cluster version age:    2 months and 27 days
##     H2O cluster name:           H2O_started_from_R_MSU-TCTO_OVCAA_mvc880
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.93 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 4.1.2 (2021-11-01)
```

Convert training data to h2o object

```
train_h2o <- as.h2o(radiomics_train)
```

Set the response column to Failure.binary

```
response <- "Failure.binary"
```

Set the predictor names

```
predictors <- setdiff(colnames(radiomics_train), response)
```

```
h2o_rf1 <- h2o.randomForest(
  x = predictors,
  y = response,
  training_frame = train_h2o,
  ntrees = n_features * 10,
  seed = 123
)
```

```
## Warning in .h2o.processResponseWarnings(res): Dropping bad and constant columns: [Institution].
## We have detected that your response column has only 2 unique values (0/1). If you wish to train a bi
```

```
h2o_rf1
```

```
## Model Details:
## ==============
##
## H2ORegressionModel: drf
## Model ID:  DRF_model_R_1671195799190_1
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1            4300                     4300              838989         3
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1        10    4.92279          5         20    10.75791
##
##
## H2ORegressionMetrics: drf
## ** Reported on training data. **
## ** Metrics reported on Out-Of-Bag training samples **
##
## MSE:  0.08314394
## RMSE:  0.2883469
## MAE:  0.209199
## RMSLE:  0.2031975
## Mean Residual Deviance :  0.08314394
```

Hyperparameter grid

```
hyper_grid <- list(
  mtries = floor(n_features * c(.05, .15, .25, .333, .4)),
  min_rows = c(1, 3, 5, 10),
  max_depth = c(10, 20, 30),
  sample_rate = c(.55, .632, .70, .80)
)
```

Random grid search strategy

```
search_criteria <- list(
  strategy = "RandomDiscrete",
  stopping_metric = "mse",
  stopping_tolerance = 0.001,
  stopping_rounds = 10,
  max_runtime_secs = 60*5
)
```

Perform grid search

```
random_grid <- h2o.grid(
  algorithm = "randomForest",
  grid_id = "rf_random_grid",
  x = predictors,
  y = response,
  training_frame = train_h2o,
  hyper_params = hyper_grid,
  ntrees = n_features * 10,
  seed = 123,
  stopping_metric = "RMSE",
  stopping_rounds = 10,
  stopping_tolerance = 0.005,
  search_criteria = search_criteria
)
```

Collect the results and sort by our model performance metric of choice

```
random_grid_perf <- h2o.getGrid(
  grid_id = "rf_random_grid",
  sort_by = "mse",
  decreasing = FALSE
)
random_grid_perf
```

```
## H2O Grid Details
## ================
##
## Grid ID: rf_random_grid
## Used hyper parameters:
##   -  max_depth
##   -  min_rows
##   -  mtries
##   -  sample_rate
## Number of models: 240
## Number of failed models: 0
##
## Hyper-Parameter Search Summary: ordered by increasing mse
##   max_depth min_rows    mtries sample_rate                model_ids     mse
## 1  30.00000  1.00000 172.00000     0.70000  rf_random_grid_model_10 0.08153
## 2  20.00000  1.00000 172.00000     0.70000 rf_random_grid_model_115 0.08153
## 3  10.00000  1.00000 172.00000     0.70000 rf_random_grid_model_145 0.08153
## 4  20.00000  5.00000 143.00000     0.80000 rf_random_grid_model_122 0.08371
## 5  10.00000  5.00000 143.00000     0.80000 rf_random_grid_model_171 0.08371
##
## ---
##     max_depth min_rows   mtries sample_rate                model_ids     mse
## 235  30.00000 10.00000 21.00000     0.80000 rf_random_grid_model_138 0.14711
## 236  20.00000 10.00000 21.00000     0.80000 rf_random_grid_model_198 0.14711
## 237  10.00000 10.00000 21.00000     0.80000 rf_random_grid_model_234 0.14711
## 238  20.00000 10.00000 21.00000     0.55000 rf_random_grid_model_134 0.15207
## 239  10.00000 10.00000 21.00000     0.55000 rf_random_grid_model_142 0.15207
## 240  30.00000 10.00000 21.00000     0.55000 rf_random_grid_model_167 0.15207
```

Re-run model with impurity-based variable importance

```
rf_impurity <- ranger(
  formula = Failure.binary ~ .,
  data = radiomics_train,
  num.trees = 2000,
  mtry = 32,
  min.node.size = 1,
  sample.fraction = .80,
  replace = FALSE,
  importance = "impurity",
  respect.unordered.factors = "order",
  verbose = FALSE,
  seed  = 123
)
```

Re-run model with permutation-based variable importance

```
rf_permutation <- ranger(
  formula = Failure.binary ~ .,
```

```
  data = radiomics_train,
  num.trees = 2000,
  mtry = 32,
  min.node.size = 1,
  sample.fraction = .80,
  replace = FALSE,
  importance = "permutation",
  respect.unordered.factors = "order",
  verbose = FALSE,
  seed  = 123
)
```

Print the top 20 features during Training

```
p1 <- vip::vip(rf_impurity, num_features = 20, bar = FALSE)
p2 <- vip::vip(rf_permutation, num_features = 20, bar = FALSE)

gridExtra::grid.arrange(p1, p2, nrow = 1)
```