INFS 692 Data Science Final Project: Model 1

Chanpreet Kaur

2022-12-16

Model 1

All code in this file is referenced from week 7 lecture class.

Task is to create an ensemble classification model with at least 3 models.

Stacked generalization or Stacking is an ensemble method where multiple base learners are trained first, and then a combiner (Super learner) is trained to make final prediction. Overall, there would be 3 steps; set up the ensemble; train the ensemble; and finally predict on new data. In my project, I shall use the Stacking as my ENSEMBLE CLASSIFICATION MODEL aka Model 1.

First, we import all required R libraries.

```r
# Helper packages
library(rsample)   # for creating our train-test splits
library(recipes)   # for minor feature engineering tasks
```

```
## Loading required package: 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

##
## Attaching package: 'recipes'

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

```r
library(ggplot2)  # for graphics

# Modeling packages
library(h2o)       # for fitting stacked models
```

```
##
## ----------------------------------------------------------------------
##
## 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
```

```r
library(rpart)    # direct engine for decision tree application
library(caret) # for classification and regression training
```

```
## Loading required package: lattice
```

```r
# Model Interpretability packages
library(vip)         # for feature importance
```

```
##
## Attaching package: 'vip'

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

```r
library(pdp)         # for feature effects/partial dependence plots

# Other packages
library(dslabs)
library(purrr)
```

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

## The following object is masked from 'package:pdp':
##
##     partial

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

```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
#library(rpart.plot)  # for plotting decision trees
```

Pre-process given data

Checking for missing values and null in data

```r
# Import data from csv file
RadData <- read.csv('./radiomics_completedata.csv')
which(is.na(RadData)) #no null or missing values in data
```

```
## integer(0)
```

2. Normalizing the data and forming a numeric data matrix

```r
institution <- RadData$Institution
Failure_binary <- RadData$Failure.binary
#Focus on numeric data;Forming a complete numeric matrix
num <- sapply(RadData, is.numeric)
RadData <- RadData[num]
Failure_binary <- RadData$Failure.binary
RadData <- Filter(function(x) !all(x %in% c(0, 1)), RadData)
f_Raddata <- scale(RadData)
f_Raddata <- as.data.frame(f_Raddata)
```

3. Fetch Correlation of variables in data

```
cor(f_Raddata)
```

Cross-validation by Splitting data into training (80%) and testing (20%)

```
set.seed(123) # for reproducibility

f_Raddata2 <- cbind(f_Raddata, Failure_binary) #combining categorical with numeric data now
f_Raddata3 <- cbind(f_Raddata2, institution)
splitData <- initial_split(f_Raddata3, prop = 0.8, strata = 'Failure') #splitting using continous value
data_train <- training(splitData)
data_test <- testing(splitData)
```

Create different training models(GLM, RF and GBM) and then stack them to form an ensemble.

Reference code from slide 68 in week7

```
# consistent categorical levels
blueprint <- recipe(Failure_binary ~ ., data = data_train) %>%
  step_other(all_nominal(), threshold = 0.005)

# h2O objects; Create training & test sets for h2o
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         48 minutes 52 seconds
##     H2O cluster timezone:       America/New_York
##     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_CHANPREET_KAUR_pxv372
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.09 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 4.2.2 (2022-10-31 ucrt)
```

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

```
##   |                                                                              |
test_h2o <- prep(blueprint, training = data_train) %>%
  bake(new_data = data_test) %>%
  as.h2o()
```

```
##   |                                                                              |
```

```
# Get response and feature names
Y <- "Failure_binary"
X <- setdiff(names(data_train), Y)

# 1st model: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",
  keep_cross_validation_predictions = TRUE, seed = 123
)
```

```
## Warning in .h2o.processResponseWarnings(res): We have detected that your response column has only 2 u
```

```
##   |                                                                              |
```

```
# 2nd model:Train & cross-validate a RF model #omitted stopping metric as it is classification
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
)
```

```
## Warning in .h2o.processResponseWarnings(res): We have detected that your response column has only 2 u
## early stopping is enabled but neither score_tree_interval or score_each_iteration are defined. Early
```

```
##   |                                                                              |
```

```
# 3rd model:Train & cross-validate a GBM model #omitted stopping metric as it is classification
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
)
```

```
## Warning in .h2o.processResponseWarnings(res): We have detected that your response column has only 2 u
## early stopping is enabled but neither score_tree_interval or score_each_iteration are defined. Early
```

```
##   |                                                                              |
```

```
# Train a stacked ensemble using all 3 models above
# reference code slide 72 week 7
ensemble <- h2o.stackedEnsemble(x = X, y = Y, training_frame = train_h2o, base_models = list(best_glm, b
```

```
## Warning in .h2o.processResponseWarnings(res): We have detected that your response column has only 2 u
```

```
##   |                                                                              |
```

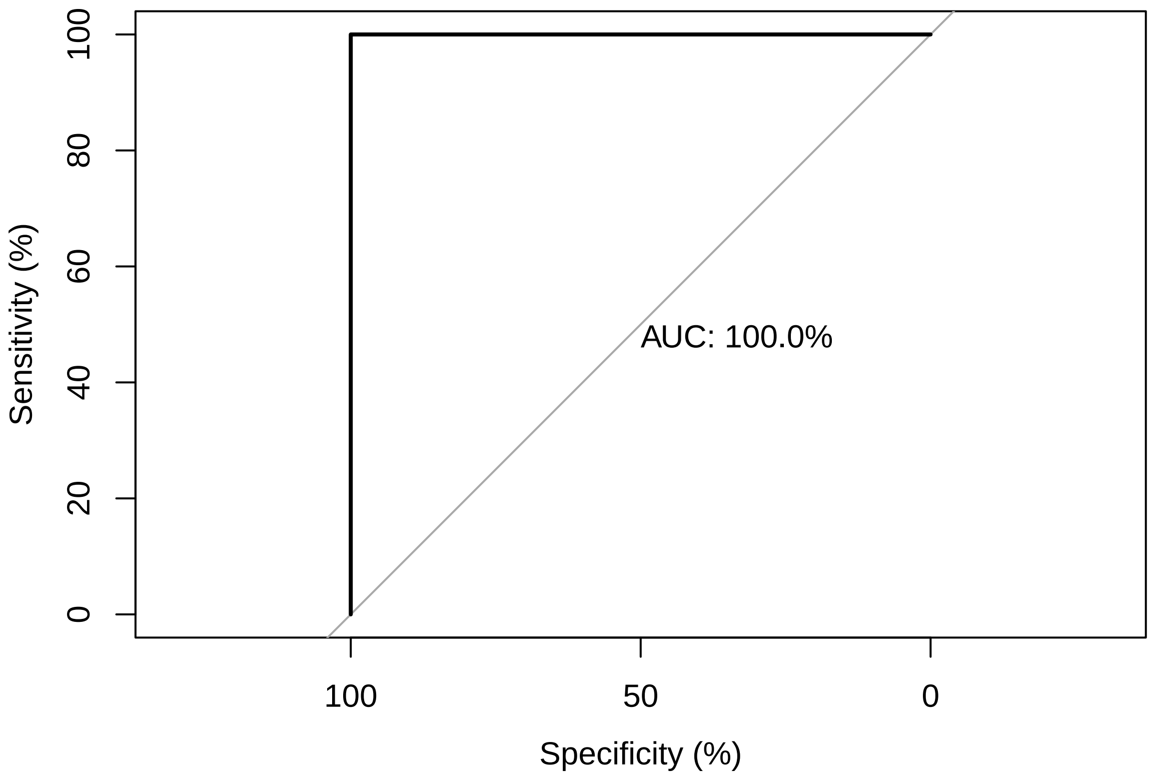
Evaluate performance during Training; AUC values during Training

```
df_train <- as.data.frame(train_h2o)
#predict probability
train_prob <- predict(ensemble, train_h2o, type = "prob")
```

```
##   |                                                                              |
```

```
df_trainprob <- as.data.frame(train_prob)

# ROC plot

roc(df_train$Failure_binary~ df_trainprob[,1], plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

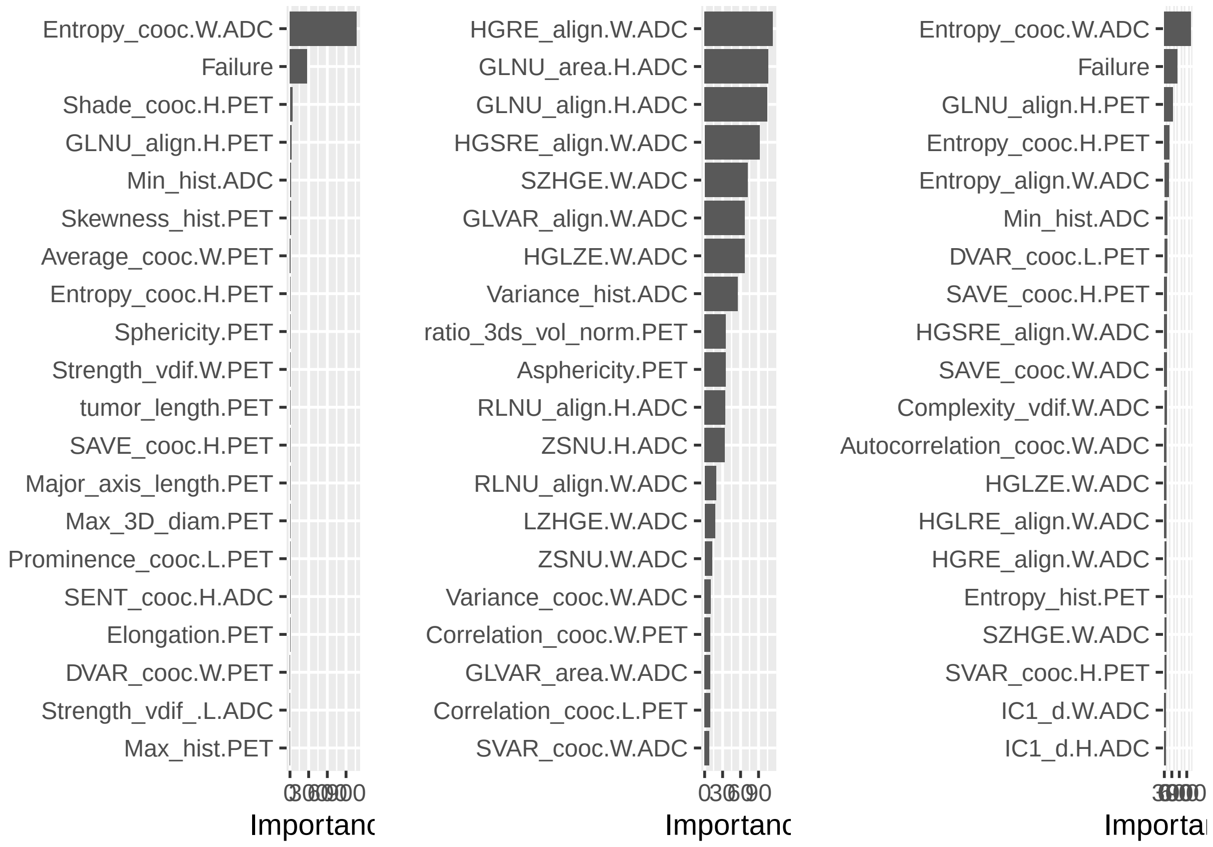
```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = df_train$Failure_binary ~ df_trainprob[,     1], plot = TRUE, legacy.axes = FA
##
## Data: df_trainprob[, 1] in 105 controls (df_train$Failure_binary 0) < 52 cases (df_train$Failure_bina
## Area under the curve: 100%
```

Feature Importance/ interpretation in training GLM, RF and GBM models; Printing the Top 20 most imp features

```
#feature importance of models
#reference code slide 28 week 7
p1<- vip::vip(best_gbm, num_features = 20, bar = FALSE)
p2<- vip::vip(best_glm, num_features = 20, bar = FALSE)
p3<- vip::vip(best_rf, num_features = 20, bar = FALSE)
gridExtra::grid.arrange(p1,p2,p3,nrow=1)
```



Evaluate performance during testing; printing the AUC values in test phase

```
# predict probabilities
df_test <- as.data.frame(test_h2o)
test_prob <- predict(ensemble, test_h2o, type = "prob")
```

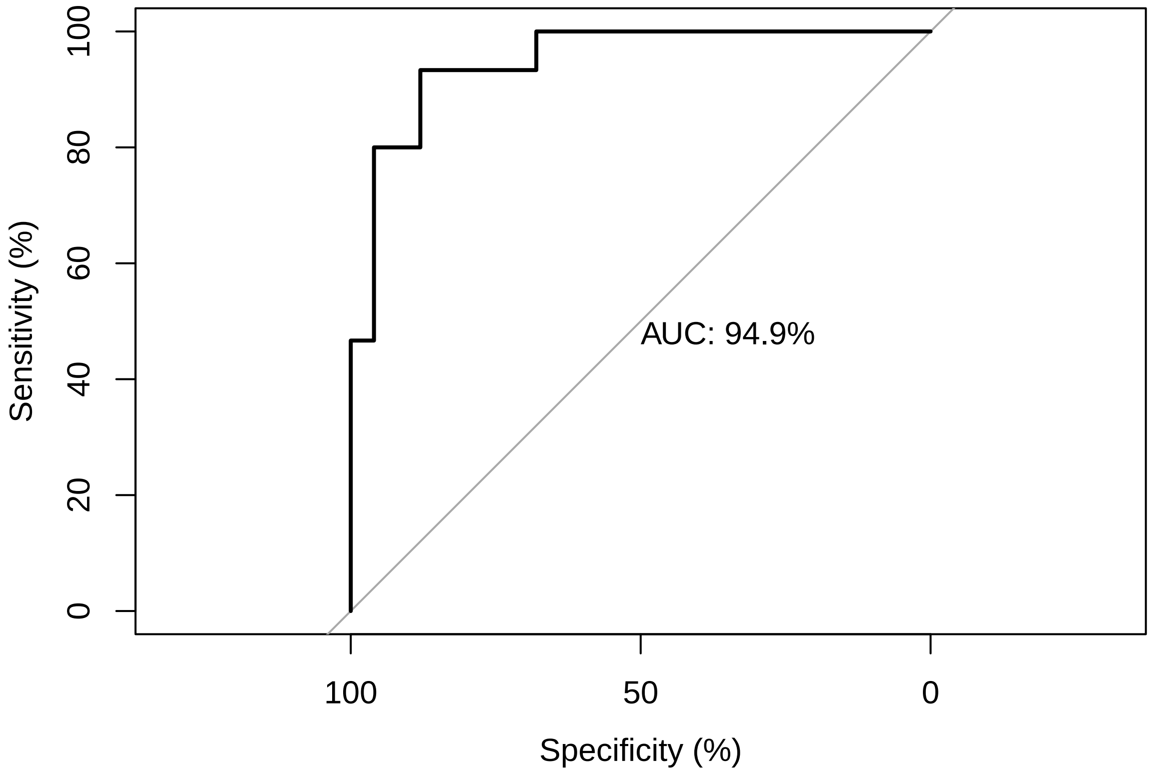
```
##   |                                                                      |
```

```
df_testprob <- as.data.frame(test_prob)

# ROC plot
```

```
roc(df_test$Failure_binary~ df_testprob[,1], plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = df_test$Failure_binary ~ df_testprob[,     1], plot = TRUE, legacy.axes = FALSI
##
## Data: df_testprob[, 1] in 25 controls (df_test$Failure_binary 0) < 15 cases (df_test$Failure_binary 
## Area under the curve: 94.93%
```