INFS692 Final Project M1

2022-12-15

Model 1

##data entry

```
library(readr)
df<- read.csv("/Users/yangyufan/Desktop/infs 692 final project/radiomics_completedata.csv")
```

Packages

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
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(ggplot2)
library(rsample)
library(keras)
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: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(rpart)
library(rpart.plot)
library(vip)
```

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

```
## The following objects are masked from 'package:keras':
##
##     metric_accuracy, metric_auc
```

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

```
library(ROCR)
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
```

Data preperation

```
data_check <- na.omit(data)
```

```
dim(df)
```

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

Split data

```
set.seed(123)
prep_df <- df %>% mutate_if(is.ordered, factor, ordered = FALSE)
df_split <- initial_split(prep_df, prop = .8, strata = "Failure.binary")
df_train <- training(df_split)
df_test  <- testing(df_split)
```

knn

```
blueprint <- recipe(Failure.binary ~ ., data = df_train) %>%
  step_other(all_nominal(), threshold = 0.005)

h2o.init()
train_h2o <- prep(blueprint, training = df_train, retain = TRUE) %>%
  juice() %>%
  as.h2o()
test_h2o <- prep(blueprint, training = df_train) %>%
  bake(new_data = df_test) %>%
  as.h2o()
```

##resampling method

```
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary)
```

Create a hyperparameter grid search

```
hyper_grid <- expand.grid(
  k = floor(seq(1, nrow(df_train)/3, length.out = 20))
)
```

Fit knn model and perform grid search

```
knn_grid <- train(
  blueprint_attr,
  data = df_train,
  method = "knn",
  trControl = cv,
  tuneGrid = hyper_grid,
  metric = "ROC"
)

ggplot(knn_grid)
```

```
varimpo <- varImp(knn_grid)
varimpo
```

print auc value during training

```
knngrid_prob <- predict(knn_grid, df_train, type = "prob")$Yes
roc(df_train$Failure.binary ~ knngrid_prob, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
title(main = "Model Performance during Training", line = 2.5)
```

top 20 importance

```
ggplot(varimpo)
```

print auc during testing

```
knngrid_probtest <- predict(knn_grid, df_test, type = "prob")$Yes
roc(df_test$Failure.binary ~ knngrid_probtest, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
title(main = "Model Performance during Testing", line = 2.5)
```

Decision Tree

auc in testing

```
tree_attri_fit <- rpart(Failure.binary~., data = df_test, method = 'class')

m1_prob <- predict(tree_attri_fit, df_test, type = "prob")

perf1 <- prediction(m1_prob[,2], df_test$failure.binary) %>%
```

```
  performance(measure = "tpr", x.measure = "fpr")
plot(perf1, col = "black", lty = 2)

roc(df_test$failure.binary ~ m1_prob[,2], plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

auc in training

```
tree_attri_fit2 <- rpart(Failure.binary~., data = df_train, method = 'class')

m2_prob <- predict(tree_attri_fit2, df_train, type = "prob")

perf2 <- prediction(m2_prob[,2], df_train$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")
plot(perf2, col = "black", lty = 2)

roc(df_train$Failure.binary ~ m2_prob[,2], plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```

logistic regression

```
dim(df)

set.seed(123)
cv_model1 <- train(
  Failure.binary ~ .,
  data = df_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

```
pred_class_1 <- predict(cv_model1, df_train)
```

```
confusionMatrix(
  data = relevel(pred_class_1, ref = "Yes"),
  reference = relevel(df_train$Failure.binary, ref = "Yes")
)
```

Compute probabilities

```
m1_prob <- predict(cv_model1, df_train, type = "prob")$Yes
```

Compute AUC metrics (training)

```
perf1 <- prediction(m1_prob, df_train$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")
title(main = "Model Performance during Training", line = 2.5)
```

Compute predicted probabilities on test data

```
m1_prob <- predict(cv_model1, df_test, type = "prob")$Yes
```

Compute AUC metrics (testing)

```
perf1 <- prediction(m1_prob, df_test$Failure.binary) %>%
  performance(measure = "tpr", x.measure = "fpr")
```

#roc

```
roc(df_test$Failure.binary ~ m1_prob, plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
title(main = "Model Performance during Testing", line = 2.5)
```