Model 1

Ying Luo

2022-12-16

All the devices that I can access to fail to run h2o. Therefore 'eval=FALSE' has to be used in order to knit the R markdown.

Helper packages

```r
library(caret)
library(rsample)
library(recipes)
library(h2o)
```

Load the dataset

```r
library(readr)
df = read.csv("radiomics_completedata.csv")
```

Investigate the statistics of the dataset

```r
summary(df)
```

Remove NA

```r
df <- na.omit(df)
```

Standardize the data

```r
df <- scale(data.matrix(df))
```

Investigate the standardized dataset

```r
df <- as.data.frame(df)
head(df)
```

Correlation of the dataset (features)

```r
corMatrix =  cor(df, y = NULL, use = "ev")
```

Split the training data and testing data by 8:2.

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

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

Get response and feature names

```r
y <- "Failure.binary"
x <- setdiff(names(churn_train), y)

churn_train[, y] <- as.factor(churn_train[, y])
churn_test[, y] <- as.factor(churn_test[, y])
```

Convert the training & test sets to an h2o object

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

Train & cross-validate a GLM model

```r
glm_model <- 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
)
```

Train & cross-validate a RF model

```r
rf_model <- h2o.randomForest(
  x = x, y = y, training_frame = train_h2o, ntrees = 1000, mtries = 20,
  max_depth = 30, min_rows = 1, sample_rate = 0.8, nfolds = 10,
  fold_assignment = "Modulo", keep_cross_validation_predictions = TRUE,
  seed = 123)
```

Train & cross-validate a GBM model

```r
gbm_model <- h2o.gbm(
  x = x, y = y, training_frame = train_h2o, ntrees = 1000, 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)
```

Train a stacked ensemble

```
ensemble <- h2o.stackedEnsemble(
  x = x, y = y, training_frame = train_h2o, base_models = list(glm_model, rf_model, gbm_model)
)
```

Print the AUC value for the ensemble model during Training

```
perf_ensemble_train <- h2o.performance(ensemble, newdata = churn_train)
ensemble_auc_train <- h2o.auc(perf_ensemble_train)
print(sprintf("Ensemble Test AUC:  %s", perf_ensemble_train))
```

Print the top 20 important features during Training

```
varimp <- h2o.varimp(ensemble)
```

Print the AUC values for the ensemble model during testing

```
perf_ensemble_test <- h2o.performance(ensemble, newdata = churn_test)
ensemble_auc_test <- h2o.auc(perf_ensemble_test)
print(sprintf("Ensemble Test AUC:  %s", perf_ensemble_test))
```

Evaluate the ensemble model performance on a test set

```
h2o.performance(ensemble, newdata = churn_test)
```

Compare to base models performance on the test set

```
perf_glm <- h2o.performance(glm_model, newdata = churn_test)
perf_rf <- h2o.performance(rf_model, newdata = churn_test)
perf_gbm <- h2o.performance(gbm_model, newdata = churn_test)

glm_auc <- h2o.auc(perf_glm)
rf_auc <- h2o.auc(perf_rf)
gbm_auc <- h2o.auc(perf_gbm)

print(sprintf("GLM Test AUC:  %s", perf_glm))
print(sprintf("GLM Test AUC:  %s", perf_rf))
print(sprintf("GLM Test AUC:  %s", perf_gbm))
```