Intuitive Package for Heterogeneous Ensemble Meta-Learning

Abstract

Combining multiple learners is an effective approach, and have been applied to many real-world problems. Ensemble learners combine a diverse collection of predictions from the individual base models to produce a composite predictive model that is more accurate and robust than its components. **metaEnsembleR** is an R package for meta-level ensemble learning (Classification, Regression) that is fully-automated. The functionalities provided includes simple user input based predictive modeling with the selection choice of the algorithms, train-validation-test split, model valuations, and easy guided unseen data prediction which can help the user's to build stack ensembles on the go. The core aim of this package is to cater the larger audiences in general. **metaEnsembleR** significantly lowers the barrier for the practitioners to apply heterogeneous ensemble learning techniques in an amateur fashion to their everyday predictive problems.

Using metaEnsembleR

The package consists of the following components:

1. Ensemble Classifiers Training and Prediction
2. Ensemble Regressor Training and Prediction
3. Model Evaluation, Model Results (Observation vs. Prediction on test data) & new unseen data prediction and Disk write I/O performance charts & saving prediction results

All these functions are very intuitive, and their use is illustrated with examples below covering the Classification and Regression problems for data with different distributions.

Usage

```
library("metaEnsembleR")
set.seed(123)
```

- Training the ensemble classification model is as simple as one-line call to the **ensembler.classifier** function, in the following ways either passing the csv file directly or the imported dataframe

```
ensembler_return ← ensembler.classifier(iris[1:130,], 5, c('treebag','rpart'), 'gbm', 0.60, 0.20, 0.20, read.csv('./unseen_data.csv'))
```

OR

```
unseen_new_data_testing ←iris[130:150,]
```

ensembler_return ← **ensembler.classifier**(iris[1:130,], 5, c('treebag','rpart'), 'gbm', 0.60, 0.20, 0.20, unseen_new_data_testing)

that takes into account the following arguments in the order

a. Dataset
b. Outcome/Response Variable index
c. Base Learners
d. Final Learner
e. Train-Validation-Test split ratio
f. Unseen data

The above function returns the following, i.e., test data with the predictions, prediction labels, model result, and finally the unseen data with the predictions.

```
testpreddata <- data.frame(ensembler_return[1])
table(testpreddata$actual_label)

table(ensembler_return[2])

####Performance comparison#####

modelresult <- ensembler_return[3]
modelresult

act_mybar <- qplot(testpreddata$actual_label, geom="bar")
act_mybar
pred_mybar <- qplot(testpreddata$predictions, geom='bar')
pred_mybar
act_tbl <- tableGrob(t(summary(testpreddata$actual_label)))
pred_tbl <- tableGrob(t(summary(testpreddata$predictions)))
ggsave("testdata_actual_vs_predicted_chart.pdf",grid.arrange(act_tbl,
pred_tbl))
ggsave("testdata_actual_vs_predicted_plot.pdf",grid.arrange(act_mybar,
pred_mybar))

####unseen data###

unseenpreddata <- data.frame(ensembler_return[4])
table(unseenpreddata$unseenpreddata)
```

- Training the ensemble regression model is the same as one-line call to the **ensembler.regression** function, in the following ways either passing the csv file directly or the imported dataframe

```
house_price <- read.csv(file = './data/regression/house_price_data.csv')
unseen_new_data_testing_house_price <- house_price[250:414,]
write.csv(unseen_new_data_testing_house_price,
'unseen_house_price_regression.csv', fileEncoding = 'UTF-8', row.names = F)
```

ensembler_return ← **ensembler.regression**(house_price[1:250,], 1, c('treebag','rpart'), 'gbm', 0.60, 0.20, 0.20, read.csv('./unseen_house_price_regression.csv'))

OR

ensembler_return ← **ensembler.regression**(house_price[1:250,], 1, c('treebag','rpart'), 'gbm', 0.60, 0.20, 0.20, unseen_new_data_testing_house_price)

that takes into account the following arguments in the order

a. Dataset
b. Outcome/Response Variable index
c. Base Learners
d. Final Learner
e. Train-Validation-Test split ratio
f. Unseen data

The above function returns the following, i.e., test data with the predictions, prediction values, model result, and finally the unseen data with the predictions.

```
testpreddata <- data.frame(ensembler_return[1])


####Performance comparison#####

modelresult <- ensembler_return[3]
modelresult
write.csv(modelresult[[1]], "performance_chart.csv")

####unseen data###

unseenpreddata <- data.frame(ensembler_return[4])
```

R session information

R version 3.5.3 (2019-03-11)

Platform: x86_64-w64-mingw32/x64 (64-bit)

Running under: Windows 10 x64 (build 18363)

locale:

[1] LC_COLLATE=English_United Kingdom.1252 LC_CTYPE=English_United Kingdom.1252 LC_MONETARY=English_United Kingdom.1252 LC_NUMERIC=C

[5] LC_TIME=English_United Kingdom.1252

attached base packages:

[1] stats graphics grDevices utils datasets methods base

other attached packages:

[1] caret_6.0-86 ggplot2_3.3.2 lattice_0.20-38 metaEnsembleR_0.1.0 gridExtra_2.3

loaded via a namespace (and not attached):

[1] gbm_2.1.5 tidyselect_1.0.0 purrr_0.3.4 reshape2_1.4.4 splines_3.5.3 colorspace_1.4-1 vctrs_0.2.4

[8] generics_0.0.2 stats4_3.5.3 survival_3.1-12 prodlim_2019.11.13 rlang_0.4.5 ModelMetrics_1.2.2.2 e1071_1.7-3

[15] pillar_1.4.6 glue_1.4.0 withr_2.3.0 foreach_1.5.0 lifecycle_0.2.0 plyr_1.8.6 lava_1.6.8

[22] stringr_1.4.0 timeDate_3043.102 munsell_0.5.0 gtable_0.3.0
recipes_0.1.13 codetools_0.2-16 labeling_0.3

[29] class_7.3-15 Rcpp_1.0.4.6 scales_1.1.1 ipred_0.9-9 farver_2.0.3
digest_0.6.25 stringi_1.4.6

[36] dplyr_0.8.5 grid_3.5.3 tools_3.5.3 magrittr_1.5 tibble_3.0.1
randomForest_4.6-14 crayon_1.3.4

[43] pkgconfig_2.0.3 ellipsis_0.3.0 MASS_7.3-51.1 Matrix_1.2-15
data.table_1.12.8 pROC_1.16.2 lubridate_1.7.8

[50] gower_0.2.1 assertthat_0.2.1 rstudioapi_0.11 iterators_1.0.12 R6_2.4.1
rpart_4.1-13 nnet_7.3-12

[57] nlme_3.1-137 compiler_3.5.3