

# Package ‘metaEnsembleR’

November 19, 2020

**Type** Package

**Title** Automated Intuitive Package for Meta-Ensemble Learning

**Version** 0.1.0

**Date** 2020-10-27

**Author** Ajay Arunachalam

**Maintainer** Ajay Arunachalam <ajay.arunachalam08@gmail.com>

**Description** Extends the base classes and methods of 'caret' package for integration of base learners. The user can input the number of different base learners, and specify the final learner, along with the train-validation-test data partition split ratio. The predictions on the unseen new data is the resultant of the ensemble meta-learning <<https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>> of the heterogeneous learners aimed to reduce the generalization error in the predictive models. It significantly lowers the barrier for the practitioners to apply heterogeneous ensemble learning techniques in an amateur fashion to their everyday predictive problems.

**License** GPL (>= 2)

**Encoding** UTF-8

**NeedsCompilation** no

**Repository** CRAN

**Imports** caret, ggplot2, graphics, e1071, gbm, randomForest

**Depends** gridExtra

**Suggests** knitr, R.rsp

**VignetteBuilder** R.rsp

**RoxygenNote** 7.1.1

**Date/Publication** 2020-11-19 09:10:05 UTC

## R topics documented:

ensembler.classifier . . . . .	2
ensembler.regression . . . . .	4

<b>Index</b>	<b>7</b>
--------------	----------

---

ensembler.classifier *Ensemble Classifiers Training & Prediction, Model Result Evaluation*

---

### Description

This function uses the base learners, and the final top layer learner to produce an ensemble prediction. The user can input the number of base learners and specify the final learner along with the train-validation-test data partition split ratio. The predictions on the unseen new data is the resultant of the ensemble meta-learning of the heterogeneous learners aimed to reduce the generalization error in the predictive models. Functions from **caret** are used for training and prediction of the base learners and the final learner.

### Usage

```
ensembler.classifier(data
  , outcomeVARIABLEINDEX
  , IndividualModels
  , TopLayerModel
  , dstr
  , dsv
  , dst
  , unseen_new_data
)
```

### Arguments

data	Data to be used for training, validation, and test.
outcomeVARIABLEINDEX	Index of the response/outcome variable.
IndividualModels	Training of base learners.
TopLayerModel	Top layer final learner.
dstr	Training data split ratio.
dsv	Validation data split ratio.
dst	Testing data split ratio.
unseen_new_data	Prediction on unseen new data.

### Value

data	Complete data or the indexed data.
outcomeVARIABLEINDEX	Integer value of the response/outcome variable index.
IndividualModels	A vector of base learners or standalone individual learner.

TopLayerModel	Name of the Top layer learner.
dstr	fraction integer. Not greater than 1. Total sum of dstr, dsv, dst should be equal to 1.
dsv	fraction integer. Not greater than 1. Total sum of dstr, dsv, dst should be equal to 1.
dst	fraction integer. Not greater than 1. Total sum of dstr, dsv, dst should be equal to 1.
unseen_new_data	Unseen new data provided either as a csv file or imported dataframe.

**Note**

The function `ensembler.classifier()`, returns a list with the following elements: `ensembler_return[1]`Test data predictions. `ensembler_return[2]`Prediction labels. `ensembler_return[3]`Model Result. `ensembler_return[4]`Unseen data predictions.

**Author(s)**

Ajay Arunachalam

**Examples**

```
library("metaEnsembleR")
attach(iris)
data("iris")
unseen_new_data_testing <- iris[130:150,]
#write.csv(unseen_new_data_testing
#         , 'unseen_check.csv'
#         , fileEncoding = 'UTF-8'
#         , row.names = FALSE)
ensembler_return <- ensembler.classifier(iris[1:130,]
, 5
, c('rpart') #c('treebag', 'rpart')
, 'rf'      # 'gbm'
, 0.60
, 0.20
, 0.20
, unseen_new_data_testing)

# or
#ensembler_return <- ensembler.classifier(iris[1:130,]
# , 5
# , c('treebag', 'rpart')
# , 'gbm'
# , 0.60
# , 0.20
# , 0.20
# , read.csv('./unseen_check.csv'))

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)

```

---

ensembler.regression    *Ensemble Regressor Training & Prediction, Model Result Evaluation*

---

## Description

This function uses the base learners, and the final top layer learner to produce an ensemble prediction. The user can input the number of base learners and specify the final learner along with the train-validation-test data partition split ratio. The predictions on the unseen new data is the resultant of the ensemble meta-learning of the heterogeneous learners aimed to reduce the generalization error in the predictive models. Functions from **caret** are used for training and prediction of the base learners and the final learner.

## Usage

```

ensembler.regression(data
  , outcomeVARIABLEINDEX
  , IndividualModels
  , TopLayerModel
  , dstr
  , dsv
  , dst
  , unseen_new_data
)

```

## Arguments

**data**                    Data to be used for training, validation, and test.

**outcomeVARIABLEINDEX**                    Index of the response/outcome variable.

IndividualModels  
 Training of base learners.  
 TopLayerModel Top layer final learner.  
 dstr Training data split ratio.  
 dsv Validation data split ratio.  
 dst Testing data split ratio.  
 unseen\_new\_data  
 Prediction on unseen new data.

**Value**

data Complete data or the indexed data.  
 outcomeVARIABLEINDEX  
 Integer value of the response/outcome variable index.  
 IndividualModels  
 A vector of base learners or standalone individual learner.  
 TopLayerModel Name of the Top layer learner.  
 dstr fraction integer. Not greater than 1. Total sum of dstr, dsv, dst should be equal to 1.  
 dsv fraction integer. Not greater than 1. Total sum of dstr, dsv, dst should be equal to 1.  
 dst fraction integer. Not greater than 1. Total sum of dstr, dsv, dst should be equal to 1.  
 unseen\_new\_data  
 Unseen new data provided either as a csv file or imported dataframe.

**Note**

The function `ensembler.regression()`, returns a list with the following elements: `ensembler_return[1]`Test data predictions. `ensembler_return[2]`Prediction values. `ensembler_return[3]`Model Result. `ensembler_return[4]`Unseen data predictions.

**Author(s)**

Ajay Arunachalam

**Examples**

```
library("metaEnsembleR")
data("rock")
unseen_rock_data <- rock[30:48,]
ensembler_return <- ensembler.regression(rock[1:30,]
, 4
, c('lm')
, 'rf'
, 0.40
```

```
      , 0.30
      , 0.30
      , unseen_rock_data)
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])
```

# Index

`ensembler.classifier`, [2](#)  
`ensembler.regression`, [4](#)