Package ‘autoBagging’

October 12, 2022

Type Package

Title Learning to Rank Bagging Workflows with Metalearning

Version 0.1.0

Author Fabio Pinto [aut],
Vitor Cerqueira [cre],
Carlos Soares [ctb],
Joao Mendes-Moreira [ctb]

Maintainer Vitor Cerqueira <cerqueira.vitormanuel@gmail.com>

Description A framework for automated machine learning. Concretely, the focus is on the optimisation of bagging workflows. A bagging workflows is composed by three phases: (i) generation: which and how many predictive models to learn; (ii) pruning: after learning a set of models, the worst ones are cut off from the ensemble; and (iii) integration: how the models are combined for predicting a new observation. autoBagging optimises these processes by combining metalearning and a learning to rank approach to learn from metadata. It automatically ranks 63 bagging workflows by exploiting past performance and dataset characterization. A complete description of the method can be found in: Pinto, F., Cerqueira, V., Soares, C., Mendes-Moreira, J. (2017): ”autoBagging: Learning to Rank Bagging Workflows with Metalearning” arXiv preprint arXiv:1706.09367.

Depends R (>= 2.10)

Imports cluster, xgboost, methods, e1071, rpart, abind, caret, MASS,
entropy, lsr, CORElearn, infotheo, minerva, party

License GPL (>= 2)

Encoding UTF-8

LazyData no

RoxygenNote 6.0.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-02 00:06:44 UTC
R topics documented:

- abmodel ......................................................... 3
- abmodel-class ............................................... 3
- autoBagging .................................................. 4
- baggedtrees .................................................. 5
- bagging ......................................................... 6
- bb .............................................................. 7
- classmajority.landmarker ......................... 7
- classmajority.landmarker.correlation .......... 8
- classmajority.landmarker.entropy .......... 8
- classmajority.landmarker.interinfo .......... 9
- classmajority.landmarker.mutual.information .9
- ContAttrs ...................................................... 10
- dstump.landmarker_d1 ...................... 10
- dstump.landmarker_d1.correlation .......... 11
- dstump.landmarker_d1.entropy .......... 11
- dstump.landmarker_d1.interinfo .......... 12
- dstump.landmarker_d1.mutual.information .12
- dstump.landmarker_d2 ...................... 13
- dstump.landmarker_d2.correlation .......... 13
- dstump.landmarker_d2.entropy .......... 14
- dstump.landmarker_d2.interinfo .......... 14
- dstump.landmarker_d2.mutual.information .15
- dstump.landmarker_d3 ...................... 15
- dstump.landmarker_d3.correlation .......... 16
- dstump.landmarker_d3.entropy .......... 16
- dstump.landmarker_d3.interinfo .......... 17
- dstump.landmarker_d3.mutual.information .17
- GetMeasure .................................................. 18
- get_target ................................................... 18
- KNORA.E ...................................................... 19
- lda.landmarker.correlation ............ 19
- majority_voting ........................................ 20
- mdsq ......................................................... 20
- nb.landmarker ............................................. 21
- nb.landmarker.correlation ............ 21
- nb.landmarker.entropy ............ 22
- nb.landmarker.interinfo ............ 22
- nb.landmarker.mutual.information ........ 23
- OLA .......................................................... 23
- predict.abmodel-method ............. 24
- ReadDF ....................................................... 24
- SymbAttrs .................................................. 25
- sysdata ..................................................... 25

Index 26
Description

abmodel

Usage

abmodel(base_models, form, data, dynamic_selection)

Arguments

base_models  a list of decision tree classifiers
form          formula
data          dataset used to train base_models
dynamic_selection  the dynamic selection/combination method to use to aggregate predictions. If none, majority vote is used.

Description

abmodel-class

abmodel-class

Description

abmodel-class

abmodel is an S4 class that contains the ensemble model. Besides the base learning algorithms–base_models–abmodel class contains information about the dynamic selection method to apply in new data.

Slots

base_models  a list of decision tree classifiers
form          formula
data          dataset used to train base_models
dynamic_selection  the dynamic selection/combination method to use to aggregate predictions. If none, majority vote is used.

See Also

autoBagging function for the method of automatic predicting of the best workflows.
Description

Learning to Rank Bagging Workflows with Metalearning

Machine Learning (ML) has been successfully applied to a wide range of domains and applications. One of the techniques behind most of these successful applications is Ensemble Learning (EL), the field of ML that gave birth to methods such as Random Forests or Boosting. The complexity of applying these techniques together with the market scarcity on ML experts, has created the need for systems that enable a fast and easy drop-in replacement for ML libraries. Automated machine learning (autoML) is the field of ML that attempts to answers these needs. Typically, these systems rely on optimization techniques such as bayesian optimization to lead the search for the best model. Our approach differs from these systems by making use of the most recent advances on metalearning and a learning to rank approach to learn from metadata. We propose autoBagging, an autoML system that automatically ranks 63 bagging workflows by exploiting past performance and dataset characterization. Results on 140 classification datasets from the OpenML platform show that autoBagging can yield better performance than the Average Rank method and achieve results that are not statistically different from an ideal model that systematically selects the best workflow for each dataset.

Usage

autoBagging(form, data)

Arguments

form formula. Currently supporting only categorical target variables (classification tasks)
data training dataset with a categorical target variable

Details

The underlying model leverages the performance of the workflows in historical data. It ranks and recommends workflows for a given classification task. A bagging workflow is comprised by the following steps:

generation the number of trees to grow
pruning the pruning of low performing trees in the ensemble
pruning cut-point a parameter of the previous step
dynamic selection the dynamic selection method used to aggregate predictions. If none is recommended, majority voting is used.

Value

an abmodel class object
References


See Also

bagging for the bagging pipeline with a specific workflow; baggedtrees for the bagging implementation; abmodel-class for the returning class object.

Examples

```r
## Not run:
# splitting an example dataset into train/test:
train <- iris[1:(0.7*nrow(iris)), ]
test <- iris[-c(1:(0.7*nrow(iris))), ]
# then apply autoBagging to the train, using the desired formula:
# autoBagging will compute metafeatures on the dataset
# and apply a pre-trained ranking model to recommend a workflow.
model <- autoBagging(Species ~., train)
# predictions are produced with the standard predict method
preds <- predict(model, test)
## End(Not run)
```

---

**baggedtrees**

*bagged trees models*

### Description

The standard resampling with replacement (bootstrap) is used as sampling strategy.

### Usage

```r
baggedtrees(form, data, ntree = 100)
```

### Arguments

- **form**: formula
- **data**: training data
- **ntree**: no of trees

### Examples

```r
ensemble <- baggedtrees(Species ~., iris, ntree = 50)
```
Description

bagging method

Usage

bagging(form, data, ntrees, pruning, dselection, pruning_cp)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>form</td>
<td>formula</td>
</tr>
<tr>
<td>data</td>
<td>training data</td>
</tr>
<tr>
<td>ntrees</td>
<td>ntrees</td>
</tr>
<tr>
<td>pruning</td>
<td>model pruning method. A character vector. Currently, the following methods are supported:</td>
</tr>
<tr>
<td></td>
<td>mdsq Margin-distance minimisation</td>
</tr>
<tr>
<td></td>
<td>bb boosting based pruning</td>
</tr>
<tr>
<td></td>
<td>none no pruning</td>
</tr>
<tr>
<td>dselection</td>
<td>dynamic selection of the available models. Currently, the following methods are supported:</td>
</tr>
<tr>
<td></td>
<td>ola Overall Local Accuracy</td>
</tr>
<tr>
<td></td>
<td>knora-e K-nearest-oracles-eliminate</td>
</tr>
<tr>
<td></td>
<td>none no dynamic selection. Majority voting is used.</td>
</tr>
<tr>
<td>pruning_cp</td>
<td>The pruning cutpoint for the pruning method picked.</td>
</tr>
</tbody>
</table>

See Also

`baggedtrees` for the implementation of the bagging model.

Examples

```r
# splitting an example dataset into train/test:
train <- iris[1:(.7*nrow(iris)), ]
test <- iris[-c(1:(.7*nrow(iris))), ]
form <- Species ~.
# a user-defined bagging workflow
m <- bagging(form, iris, ntrees = 5, pruning = "bb", pruning_cp = .5, dselection = "ola")
preds <- predict(m, test)
# a standard bagging workflow with 5 trees (5 trees for examplification purposes):
m2 <- bagging(form, iris, ntrees = 5, pruning = "none", dselection = "none")
preds2 <- predict(m2, test)
```
**bb**

*Boosting-based pruning of models*

**Description**

Boosting-based pruning of models

**Usage**

`bb(form, preds, data, cutPoint)`

**Arguments**

- `form` formula
- `preds` predictions in training data
- `data` training data
- `cutPoint` ratio of the total number of models to cut off

---

**classmajority.landmarker**

*classmajority.landmarker*

**Description**

classmajority.landmarker

**Usage**

`classmajority.landmarker(dataset, data.char)`

**Arguments**

- `dataset` train data for the landmarker
- `data.char` dc
**classmajority.landmarker.correlation**

*Description*

classmajority.landmarker.correlation

*Usage*

`classmajority.landmarker.correlation(dataset, data.char)`

*Arguments*

- `dataset`: train data for the landmarker
- `data.char`: dc

---

**classmajority.landmarker.entropy**

*Description*

classmajority.landmarker.entropy

*Usage*

`classmajority.landmarker.entropy(dataset, data.char)`

*Arguments*

- `dataset`: train data for the landmarker
- `data.char`: dc
classmajority.landmarker.interinfo

**Description**

classmajority.landmarker.interinfo

**Usage**

classmajority.landmarker.interinfo(dataset, data.char)

**Arguments**

- **dataset**: train data for the landmarker
- **data.char**: dc

classmajority.landmarker.mutual.information

**Description**

classmajority.landmarker.mutual.information

**Usage**

classmajority.landmarker.mutual.information(dataset, data.char)

**Arguments**

- **dataset**: train data for the landmarker
- **data.char**: dc
ContAttr

Retrieve names of continuous attributes (not including the target)

Description

Retrieve names of continuous attributes (not including the target)

Usage

ContAttrs(dataset)

Arguments

dataset structure describing the data set, according to read_data.R

Value

list of strings

See Also

read_data.R

dstump.landmarker_d1

dstump.landmarker_d1

Description

dstump.landmarker_d1

Usage

dstump.landmarker_d1(dataset, data.char)

Arguments

dataset train data for the landmarker

data.char dc
\textbf{dstump.landmarker\_d1.correlation}

\textbf{Description}
\begin{quote}
dstump.landmarker\_d1.correlation
\end{quote}

\textbf{Usage}
\begin{quote}
dstump.landmarker\_d1.correlation(dataset, data.char)
\end{quote}

\textbf{Arguments}
\begin{quote}
\begin{tabular}{ll}
dataset & train data for the landmarker \\
data.char & dc \\
\end{tabular}
\end{quote}

\textbf{dstump.landmarker\_d1.entropy}

\textbf{Description}
\begin{quote}
dstump.landmarker\_d1.entropy
\end{quote}

\textbf{Usage}
\begin{quote}
dstump.landmarker\_d1.entropy(dataset, data.char)
\end{quote}

\textbf{Arguments}
\begin{quote}
\begin{tabular}{ll}
dataset & train data for the landmarker \\
data.char & dc \\
\end{tabular}
\end{quote}
Description

dstump.landmarker_d1.interinfo

Usage

dstump.landmarker_d1.interinfo(dataset, data.char)

Arguments

dataset : train data for the landmarker

data.char : dc

Description

dstump.landmarker_d1.mutual.information

Usage

dstump.landmarker_d1.mutual.information(dataset, data.char)

Arguments

dataset : train data for the landmarker

data.char : dc
**dstump.landmarker_d2**

---

**Description**

dstump.landmarker_d2

**Usage**

```r
dstump.landmarker_d2(dataset, data.char)
```

**Arguments**

- **dataset**: train data for the landmarker
- **data.char**: dc

---

**dstump.landmarker_d2.correlation**

---

**Description**

dstump.landmarker_d2.correlation

**Usage**

```r
dstump.landmarker_d2.correlation(dataset, data.char)
```

**Arguments**

- **dataset**: train data for the landmarker
- **data.char**: dc
**dstump.landmarker_d2.entropy**

**Description**

**Usage**

```r
dstump.landmarker_d2.entropy(dataset, data.char)
```

**Arguments**

- `dataset` : train data for the landmarker
- `data.char` : dc

---

**dstump.landmarker_d2.interinfo**

**Description**

**Usage**

```r
dstump.landmarker_d2.interinfo(dataset, data.char)
```

**Arguments**

- `dataset` : train data for the landmarker
- `data.char` : dc
Description
dstump.landmarker_d2.mutual.information

Usage
dstump.landmarker_d2.mutual.information(dataset, data.char)

Arguments
dataset train data for the landmarker
data.char dc

Description
dstump.landmarker_d3

Usage
dstump.landmarker_d3(dataset, data.char)

Arguments
dataset train data for the landmarker
data.char dc
Description

dstump.landmarker_d3.correlation

Usage

dstump.landmarker_d3.correlation(dataset, data.char)

Arguments

dataset: train data for the landmarker
data.char: dc

Description

dstump.landmarker_d3.entropy

Usage

dstump.landmarker_d3.entropy(dataset, data.char)

Arguments

dataset: train data for the landmarker
data.char: dc
Description

dstump.landmarker_d3.interinfo

Usage

dstump.landmarker_d3.interinfo(dataset, data.char)

Arguments

dataset 
  train data for the landmarker

data.char 
  dc

Description

dstump.landmarker_d3.mutual.information

Usage

dstump.landmarker_d3.mutual.information(dataset, data.char)

Arguments

dataset 
  train data for the landmarker

data.char 
  dc
GetMeasure

Retrieve the value of a previously computed measure

Description
Retrieve the value of a previously computed measure

Usage
GetMeasure(inDCName, inDCSet, component.name = "value")

Arguments
inDCName     name of data characteristics
inDCSet      set of data characteristics already computed
component.name name of component (e.g. time or value) to retrieve; if NULL retrieve all

Value
simple or structured value

Note
if measure is not available, stop execution with error

get_target

get target variable

Description
get the target variable from a formula

Usage
get_target(form)

Arguments
form         formula
KNORA.E  

**K-Nearest-ORAcle-Eliminate**

**Description**

A dynamic selection method

**Usage**

```r
KNORA.E(form, mod, v.data, t.data, k = 5)
```

**Arguments**

- `form` : formula
- `mod` : a list comprising the individual models
- `v.data` : validation data
- `t.data` : test data, with the instances to predict
- `k` : the number of nearest neighbors. Defaults to 5.

---

**lda.landmarker.correlation**

**ldalandmarker.correlation**

**Description**

lda.landmarker.correlation

**Usage**

```r
## S3 method for class 'landmarker.correlation'
lda(dataset, data.char)
```

**Arguments**

- `dataset` : train data for the landmarker
- `data.char` : dc
### majority_voting

**Description**

majority voting

**Usage**

`majority_voting(x)`

**Arguments**

- `x`: predictions produced by a set of models

### mdsq

**Description**

Margin Distance Minimization

**Usage**

`mdsq(form, preds, data, cutPoint)`

**Arguments**

- `form`: formula
- `preds`: predictions in training data
- `data`: training data
- `cutPoint`: ratio of the total number of models to cut off
**nb.landmarker**

**Description**

nb.landmarker

**Usage**

`nb.landmarker(dataset, data.char)`

**Arguments**

- **dataset**: train data for the landmarker
- **data.char**: dc

---

**nb.landmarker.correlation**

**Description**

nb.landmarker.correlation

**Usage**

`nb.landmarker.correlation(dataset, data.char)`

**Arguments**

- **dataset**: train data for the landmarker
- **data.char**: dc
Description

nb.landmarker.entropy

Usage

nb.landmarker.entropy(dataset, data.char)

Arguments

dataset: train data for the landmarker

data.char: dc

Description

nb.landmarker.interinfo

Usage

nb.landmarker.interinfo(dataset, data.char)

Arguments

dataset: train data for the landmarker

data.char: dc
Description

nb.landmarker.mutual.information

Usage

nb.landmarker.mutual.information(dataset, data.char)

Arguments

dataset train data for the landmarker
data.char dc

OLA Overall Local Accuracy

Description

A dynamic selection method

Usage

OLA(form, mod, v.data, t.data, k = 5)

Arguments

form formula
mod a list comprising the individual models
v.data validation data
t.data test data, with the instances to predict
k the number of nearest neighbors. Defaults to 5.
predict,abmodel-method

*Predicting on new data with a **abmodel** model*

---

**Description**

This is a predict method for predicting new data points using a **abmodel** class object - referring to an ensemble of bagged trees

**Usage**

```r
## S4 method for signature 'abmodel'
predict(object, newdata)
```

**Arguments**

- `object`: A **abmodel-class** object.
- `newdata`: New data to predict using an **abmodel** object

**Value**

Predictions produced by an **abmodel** model.

**See Also**

- **abmodel-class** for details about the bagging model:

---

**ReadDF**

*FUNCTION TO TRANSFORM DATA FRAME INTO LIST WITH GSI REQUIREMENTS*

---

**Description**

FUNCTION TO TRANSFORM DATA FRAME INTO LIST WITH GSI REQUIREMENTS

**Usage**

```r
ReadDF(dat)
```

**Arguments**

- `dat`: data frame

**Value**

A list containing components that describe the names (see `ReadtAttrsInfo`) and the data (see `ReadData`) files

THIS FUNCTION HAS TO BE BASED IN `READATTRSINFO` AND `READDATA`
SymbAttrs

Retrieve names of symbolic attributes (not including the target)

Description
Retrieve names of symbolic attributes (not including the target)

Usage
SymbAttrs(dataset)

Arguments
dataset structure describing the data set, according to read_data.R

Value
list of strings

See Also
read_data.R

sysdata

Meta data needed to run the autoBagging method.

Description
Meta data needed to run the autoBagging method.

Usage
sysdata

Format
a list comprising the following information

avgRankMatrix the average rank data regarding each bagging workflow
workflows metadata on the bagging workflows
MaxMinMetafeatures range data on each metafeature
metafeatures names and values of each metafeatures used to describe the datasets
metamodel the xgboost ranking metamodel
Index

* datasets
  sysdata, 25
  abmodel, 3
  abmodel-class, 3
  autoBagging, 3, 4
  autoBagging-package (autoBagging), 4
  baggedtrees, 5, 6
  bagging, 5, 6
  bb, 7
  classmajority.landmarker, 7
  classmajority.landmarker.correlation, 8
  classmajority.landmarker.entropy, 8
  classmajority.landmarker.interinfo, 9
  classmajority.landmarker.mutual.information, 9
  ContAttrs, 10
  dstump.landmarker_d1, 10
  dstump.landmarker_d1.correlation, 11
  dstump.landmarker_d1.entropy, 11
  dstump.landmarker_d1.interinfo, 12
  dstump.landmarker_d1.mutual.information, 12
  dstump.landmarker_d2, 13
  dstump.landmarker_d2.correlation, 13
  dstump.landmarker_d2.entropy, 14
  dstump.landmarker_d2.interinfo, 14
  dstump.landmarker_d2.mutual.information, 15
  dstump.landmarker_d3, 15
  dstump.landmarker_d3.correlation, 16
  dstump.landmarker_d3.entropy, 16
  dstump.landmarker_d3.interinfo, 17
  dstump.landmarker_d3.mutual.information, 17
  get_target, 18

GetMeasure, 18

KNORA.E, 19

lda.landmarker.correlation, 19

majority_voting, 20

mdsq, 20

nb.landmarker, 21

nb.landmarker.correlation, 21

nb.landmarker.entropy, 22

nb.landmarker.interinfo, 22

nb.landmarker.mutual.information, 23

OLA, 23

predict, abmodel-method, 24

ReadDF, 24

SymbAttrs, 25

sysdata, 25