Package ‘GPLTR’

August 26, 2023

Type Package
Title Generalized Partially Linear Tree-Based Regression Model
Version 1.4
Date 2023-08-26
Author Cyprien Mbogning <cyprien.mbogning@inserm.fr> and Wilson Toussile
Maintainer Cyprien Mbogning <cyprien.mbogning@gmail.com>
Description Combining a generalized linear model with an additional tree part on the same scale. A four-step procedure is proposed to fit the model and test the joint effect of the selected tree part while adjusting on confounding factors. We also proposed an ensemble procedure based on the bagging to improve prediction accuracy and computed several scores of importance for variable selection. See ‘Cyprien Mbogning et al.’ (2014) <doi:10.1186/2043-9113-4-6> and ‘Cyprien Mbogning et al.’ (2015) <doi:10.1159/000380850> for an overview of all the methods implemented in this package.
License GPL (>= 2.0)
LazyLoad yes
Depends rpart, parallel
NeedsCompilation no
Repository CRAN
Date/Publication 2023-08-26 17:50:02 UTC

R topics documented:

GPLTR-package .......................................................... 2
bag.aucoob .............................................................. 5
bagging.pltr ............................................................ 6
best.tree.BIC.AIC ...................................................... 9
best.tree.bootstrap ...................................................... 11
best.tree.CV ........................................................... 13
best.tree.permute ....................................................... 15
burn ........................................................................... 17
data_pltr ................................................................. 18
GPLTR-package

Fit a generalized partially linear tree-based regression model

Description

Combining a generalized linear model with an additional tree part on the same scale. A four-step procedure is proposed to fit the model and test the joint effect of the selected tree part while adjusting on confounding factors. We also proposed an ensemble procedure based on the bagging to improve prediction accuracy and computed several scores of importance for variable selection. See 'Cyprien Mbogning et al.'(2014)<doi:10.1186/2043-9113-4-6>, 'Cyprien Mbogning et al.'(2015)<doi:10.1159/000380850> for an overview of all the methods implemented in this package.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>GPLTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.4</td>
</tr>
<tr>
<td>Date</td>
<td>2023-08-26</td>
</tr>
<tr>
<td>License</td>
<td>GPL(&gt;=2.0)</td>
</tr>
</tbody>
</table>

Author(s)

Cyprien Mbogning and Wilson Toussile
Maintainer: Cyprien Mbogning <cyprien.mbogning@gmail.com>

References


Examples

```r
### Example on a public dataset: the burn data
### The burn data are also displayed in the KMsurv package
### Not run:
data(burn)

### Build the rpart tree with all the variables
rpart.burn <- rpart(D2 ~ Z1 + Z2 + Z3 + Z4 + Z5 + Z6 + Z7 + Z8 + Z9 + Z10 + Z11, data = burn, method = "class")
plot(rpart.burn, main = 'rpart tree')
text(rpart.burn, xpd = TRUE, cex = .6, use.n = TRUE)

### fit the PLTR model after adjusting on gender (Z2) using the proposed method
args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxcompete = 0, maxsurrogate = 0)
family <- "binomial"
X.names = "Z2"
Y.name = "D2"
G.names = c("Z1", "Z3", "Z4", "Z5", "Z6", "Z7", "Z8", "Z9", "Z10", "Z11")
pltr.burn <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart, family = family, iterMax = 4, iterMin = 3, verbose = FALSE)

### Prunned back the maximal tree using either the BIC or the AIC criterion
pltr.burn_prun <- best.tree.BIC.AIC(xtree = pltr.burn$tree, burn, Y.name, X.names, family = family)

### plot the BIC selected tree
plot(pltr.burn_prun$tree$BIC, main = 'BIC selected tree')
text(pltr.burn_prun$tree$BIC, xpd = TRUE, cex = .6, col = 'blue')

### Summary of the selected tree by a BIC criterion
summary(pltr.burn_prun$tree$BIC)

### Summary of the final selected pltr model
```
summary(pltr.burn_prun$fit_glm$BIC)

## fit the PLTR model after adjusting on gender (Z2) using the parametric
## bootstrap method

## set numWorkers = 1 on a windows platfrom

args.parallel = list(numWorkers = 10)

best_bootstrap <- best.tree.bootstrap(pltr.burn$tree, burn, Y.name, X.names,
    G.names, B = 2000, BB = 2000, args.rpart = args.rpart, epsi = 0.008,
    iterMax = 6, iterMin = 5, family = family, LEVEL = 0.05, LB = FALSE,
    args.parallel = args.parallel, verbose = FALSE)

plot(best_bootstrap$selected_model$tree, main = 'original method')

text(best_bootstrap$selected_model$tree, xpd = TRUE)

## Bagging a set of basic unprunned pltr predictors
# ?bagging.pltr

Bag.burn <- bagging.pltr(burn, Y.name, X.names, G.names, family,
    args.rpart, epsi = 0.01, iterMax = 4, iterMin = 3,
    Bag = 10, verbose = FALSE, doprune = FALSE)

## The threshold values used

Bag.burn$CUT

## The set of PLTR models in the bagging procedure

PLTR_BAG.burn <- Bag.burn$Glm_BAG

## The set of trees in the bagging procedure

TREE_BAG.burn <- Bag.burn$Tree_BAG

## Use the bagging procedure to predict new features
# ?predict_bagg.pltr

Pred_Bag.burn <- predict_bagg.pltr(Bag.burn, Y.name, newdata = burn,
    type = "response", threshold = seq(0, 1, by = 0.1))

## The confusion matrix for each threshold value using the majority vote

Pred_Bag.burn$CONF1

## The prediction error for each threshold value

Pred_Bag.burn$PRED_ERROR1

## Compute the variable importances using the bagging procedure
Var_Impl_BAG.burn <- VIMBAG(Bag.burn, burn, Y.name)

## Importance score using the permutation method for each threshold value

Var_Impl_BAG.burn$PIS

## Shadow plot of three proposed scores

par(mfrow=c(1,3))
barplot(Var_Impl_BAG.burn$PIS$CUT5, main = 'PIS', horiz = TRUE, las = 1, cex.names = .8, col = 'lightblue')
barplot(Var_Impl_BAG.burn$DIS, main = 'DIS', horiz = TRUE, las = 1, cex.names = .8, col = 'grey')
barplot(Var_Impl_BAG.burn$DDIS, main = 'DDIS', horiz = TRUE, las = 1, cex.names = .8, col = 'purple')

## End(Not run)

---

**bag.aucoob**  
*AUC on the Out Of Bag samples*

### Description

Compute the AUC on the OOB samples of the bagging procedure for the binomial family. The true and false positive rates are also returned and could be helpful for plotting the ROC curves.

### Usage

```r
bag.aucoob(bag_pltr, xdata, Y.name)
```

### Arguments

- `bag_pltr`: The output of the function `bagging.pltr`  
- `xdata`: The learning dataset containing the dependent variable, the confounding variables and the predictors variables  
- `Y.name`: The name of the binary dependent variable

### Details

The threshold values used for computing the AUC are defined when building the bagging predictor. See `bagging.pltr` for the convenient parameterization.

### Value

A list of 4 elements

- `AUCOOB`: the AUC computed on OOB samples of the Bagging procedure  
- `TPR`: the true positive rate for several threshold values  
- `FPR`: the false positive rate for several threshold values  
- `OOB`: the Out Of Bag error for each threshold value
Note

The plot of the ROC curve is straightforward using the TPR and FPR obtained with the function bag.aucoob

Author(s)

Cyprien Mbogning

References


Examples

```r
##

bagging.pltr

bagging pltr models

Description

bagging procedure to aggregate several PLTR models for accurate prediction and variable selection

Usage

`bagging.pltr(xdata, Y.name, X.names, G.names, family = "binomial", args.rpart, epsi = 0.001, iterMax = 5, iterMin = 3, LB = FALSE, args.parallel = list(numWorkers = 1), Bag = 20, Pred_Data = data.frame(), verbose = TRUE, doprune = FALSE, thresshold = seq(0, 1, by = 0.1))`

Arguments

- **xdata**: the learning data frame
- **Y.name**: the name of the binary dependent variable
- **X.names**: the names of independent variables to consider in the linear part of the glm and as offset in the tree part
- **G.names**: the names of independent variables to consider in the tree part of the hybrid glm.
- **family**: the glm family considered depending on the type of the dependent variable (only the binomial family works in this function for the moment). a list of options that control details of the rpart algorithm. minbucket: the minimum number of observations in any terminal <leaf> node; cp: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); maxdepth: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
**epsit**
a treshold value to check the convergence of the algorithm

**iterMax**
the maximal number of iteration to consider

**iterMin**
the minimum number of iteration to consider

**LB**
a binary indicator with values TRUE or FALSE indicating whether the loading is balanced or not in the parallel computing. It is nevertheless useless on a windows platform. See mclapply

**args.parallel**
a list of two elements containing the number of workers and the type of parallelization to achieve see mclapply.

**Bag**
The number of Bagging samples to consider

**Pred_Data**
An optional data frame to validate the bagging procedure (the test dataset)

**verbose**
Logical; TRUE for printing progress during the computation (helpful for debugging)

**doprune**
a binary indicator with values TRUE or FALSE indicating whether the set of trees in the bagging procedure are pruned (by a BIC procedure) or not

**threshold**
a vector of numerical values between 0 and 1 used as threshold values for the computation of the OOB error rate

**Details**
For the Bagging procedure, it is mandatory to set maxcompete = 0 and maxsurrogate = 0 within the rpart arguments. This will ensure the correct calculation of the importance of variables.

**Value**
A list with eleven elements

**IND_OOB**
A list of length Bag containing the Out Of Bag (OOB) individuals for each PLTR model.

**EOOB**
The vector of OOB errors of the bagging procedure for each threshold value.

**OOB_ERRORS_PBP**
A matrix with Bag columns and threshold rows containing OOB error of each PLTR model in the bagging sequence for each threshold value.

**OOB_ERROR_PBP**
A vector containing the mean of OOB_ERRORS_PBP for each threshold value.

**Tree_BAG**
A list of length Bag containing the bagging trees

**Glm_BAG**
A list of length Bag containing the bagging pltr model; could be helpful for prediction on new features.

**LOST**
The 0, 1 lost matrix for OOB observations at each threshold value

**TEST**
A value of NULL if Pred_Data is not available. A list of three elements otherwise: PRED_ERROR: the estimated error of the Bagging procedure on the test sample for each threshold value; PRED_IND: A list of length the length of the threshold vector, each element containing a matrix with the prediction of the testing data individuals using each PLTR model of the bagging sequence (column by column); FINAL_PRED_IND: A list containing the final prediction of each individual of the testing data by the bagging procedure (the modal prediction) for each threshold value.
Var_IMP A numeric vector containing the relative variable importance of the bagging procedure
Timediff The execution time of the bagging procedure
CUT The threshold value used inside the bagging procedure

Author(s)
Cyprien Mbogning

References

See Also
predict_bag.pltr

Examples
## Not run:
## load the data set
data(burn)

## set the parameters
args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxsurrogate = 0)
family <- "binomial"
Y.name <- "D2"
X.names <- "Z2"
G.names <- c("Z1","Z3","Z4","Z5","Z6","Z7","Z8","Z9","Z10","Z11")
args.parallel = list(numWorkers = 1)

## Bagging a set of basic unprunned pltr predictors
Bag.burn <- bagging.pltr(burn, Y.name, X.names, G.names, family,
args.rpart, epsi = 0.01, iterMax = 4, iterMin = 3,
Bag = 20, verbose = FALSE, doprune = FALSE)

## End(Not run)
**Description**

this function is set to prune back the maximal tree by using the BIC or the AIC criterion.

**Usage**

```r
best.tree.BIC.AIC(xtree, xdata, Y.name, X.names,
                   family = "binomial", verbose = TRUE)
```

**Arguments**

- `xtree`: a tree to prune
- `xdata`: the dataset used to build the tree
- `Y.name`: the name of the dependent variable
- `X.names`: the names of independent confounding variables to consider in the linear part of the `glm`
- `family`: the `glm` family considered depending on the type of the dependent variable.
- `verbose`: Logical; TRUE for printing progress during the computation (helpful for debugging)

**Value**

a list of four elements:

- `best_index`: The size of the selected trees by BIC and AIC
- `tree`: The selected trees by BIC and AIC
- `fit_glm`: The fitted pltr models selected with BIC, and AIC
- `Timediff`: The execution time of the selection procedure

**Author(s)**

Cyprien Mbogning and Wilson Toussile

**References**


See Also

best.tree.CV, pltr.glm

Examples

data(burn)

args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxcompete = 0,
                   maxsurrogate = 0)
family <- "binomial"
X.names = "Z2"
Y.name = "D2"
G.names = c('Z1','Z3','Z4','Z5','Z6','Z7','Z8','Z9','Z10','Z11')

pltr.burn <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart,
                       family = family, iterMax = 4, iterMin = 3, verbose = FALSE)

## Prunned back the maximal tree using either the BIC or the AIC criterion
pltr.burn_prun <- best.tree.BIC.AIC(xtree = pltr.burn$tree, burn, Y.name,
                                      X.names, family = family)

## plot the BIC selected tree
plot(pltr.burn_prun$tree$BIC, main = 'BIC selected tree')
text(pltr.burn_prun$tree$BIC, xpd = TRUE, cex = .6, col = 'blue')

## Not run:
#load the data set
data(data_pltr)

## Set the parameters
args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree
fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
                      family = family, iterMax = 5, iterMin = 3)

## prunned back the maximal tree by BIC or AIC criterion

## Not run:
tree_select <- best.tree.BIC.AIC(xtree = fit_pltr$tree, data_pltr,Y.name,
                                  X.names, family = family)
plot(tree_select$tree$BIC, main = 'BIC TREE')
text(tree_select$tree$BIC, minlength = 0L, xpd = TRUE, cex = .6)
best.tree.bootstrap

## End(Not run)

---

**best.tree.bootstrap**

*parametric bootstrap on a pltr model*

### Description

A parametric bootstrap procedure to select and test at the same time the selected tree.

### Usage

```
best.tree.bootstrap(xtree, xdata, Y.name, X.names, G.names, B = 10, BB = 10,
args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10), epsi = 0.001,
iterMax = 5, iterMin = 3, family = "binomial", LEVEL = 0.05, LB = FALSE,
args.parallel = list(numWorkers = 1), verbose = TRUE)
```

### Arguments

- **xtree**: the maximal tree obtained by the function `pltr.glm`
- **xdata**: the data frame used to build `xtree`
- **Y.name**: the name of the dependent variable
- **X.names**: the names of independent variables to consider in the linear part of the glm
- **G.names**: the names of independent variables to consider in the tree part of the hybrid glm.
- **B**: the size of the bootstrap sample
- **BB**: the size of the bootstrap sample to compute the adjusted p-value
- **args.rpart**: a list of options that control details of the `rpart` algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
- **epsi**: a threshold value to check the convergence of the algorithm
- **iterMax**: the maximal number of iteration to consider
- **iterMin**: the minimum number of iteration to consider
- **family**: the glm family considered depending on the type of the dependent variable.
- **LEVEL**: the level of the test
- **LB**: a binary indicator with values `TRUE` or `FALSE` indicating whether the loading is balanced or not in the parallel computing. It is useless on a windows platform.
- **args.parallel**: parameters of the parallelization. See `mclapply` for more details
- **verbose**: Logical; `TRUE` for printing progress during the computation (helpful for debugging)
Value

- Selected_model: a list with the fit of the selected pltr model `fit_glm`, the selected tree `tree`, the p-value of the selected tree `p.value`, the adjusted p-value of the selected tree `adj_p.value`, and an indicator `Tree_Selected` to assess whether the test is significant or not.
- `fit_glm`: the fitted pltr model under the null hypothesis if the test is not significant
- `Timediff`: The execution time of the parametric bootstrap procedure
- `comp_p_values`: The P-values of the competing trees
- `Badj`: The number of samples used in the inner level of the procedure
- `BBadj`: The number of samples used in the outer level of the procedure

Author(s)

Cyprien Mbogning and Wilson Toussile

References


See Also

- `p.val.tree`

Examples

```r
# load the data set
data(data_pltr)
args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")
## Not run:
## build a maximal tree
fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names,
  args.rpart = args.rpart, family = family, iterMax = 5, iterMin = 3)

## select an test the selected tree by a parametric bootstrap procedure
args.parallel = list(numWorkers = 1, type = "PSOCK")
best_bootstrap <- best.tree.bootstrap(fit_pltr$tree, data_pltr, Y.name, X.names,
  G.names, B = 10, BB = 10, args.rpart = args.rpart, epsi = 0.001,
  iterMax = 5, iterMin = 3, family = family, LEVEL = 0.05, LB = FALSE,
  args.parallel = args.parallel)
```
## End(Not run)

**best.tree.CV**

### Pruning the Maximal tree

**Description**

This function is set to prune back the maximal tree by using a K-fold cross-validation procedure.

**Usage**

```r
best.tree.CV(xtree, xdata, Y.name, X.names, G.names, family = "binomial",
args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10), epsi = 0.001,
iterMax = 5, iterMin = 3, ncv = 10, verbose = TRUE)
```

**Arguments**

- `xtree`: a tree to prune
- `xdata`: the dataset used to build the tree
- `Y.name`: the name of the dependent variable
- `X.names`: the names of independent variables to consider in the linear part of the glm
- `G.names`: the names of independent variables to consider in the tree part of the hybrid glm.
- `family`: the glm family considered depending on the type of the dependent variable.
- `args.rpart`: a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
- `epsi`: a threshold value to check the convergence of the algorithm
- `iterMax`: the maximal number of iteration to consider
- `iterMin`: the minimum number of iteration to consider
- `ncv`: The number of folds to consider for the cross-validation
- `verbose`: Logical; TRUE for printing progress during the computation (helpful for debugging)

**Value**

A list of five elements:

- `best_index`: The size of the selected tree by the cross-validation procedure
- `tree`: The selected tree by CV
- `fit_glm`: The fitted gpltr models selected with CV
CV_ERRORS  A list of two elements containing the cross-validation error of the selected tree by the CV procedure and a vector of cross-validation errors of all the competing models

Timediff  The execution time of the Cross-Validation procedure

Author(s)
Cyprien Mbogning

References

See Also

best.tree.BIC.AIC, pltr.glm

Examples

## Not run:
## load the data set

data(data_pltr)

## set the parameters

args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree

fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
                      family = family, iterMax = 5, iterMin = 3)

## prunned back the maximal tree by a cross-validation procedure

tree_selected <- best.tree.CV(fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
                              family = family, args.rpart = args.rpart, epsi = 0.001, iterMax = 5,
                              iterMin = 3, ncv = 10)

plot(tree_selected$tree, main = 'CV TREE')
text(tree_selected$tree, minlength = 0L, xpd = TRUE, cex = .6)

## End(Not run)
**Description**

A unified permutation test procedure to select and test at the same time the selected tree

**Usage**

```r
best.tree.permute(xtree, xdata, Y.name, X.names, G.names, B = 10,
args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10), epsi = 0.001,
iterMax = 5, iterMin = 3, family = “binomial”, LEVEL = 0.05,
LB = FALSE, args.parallel = list(numWorkers = 1, type = “PSOCK”), verbose = TRUE)
```

**Arguments**

- `xtree`: The maximal tree obtained by the function `pltr.glm`
- `xdata`: The data frame used to build `xtree`
- `Y.name`: The name of the dependent variable
- `X.names`: The names of independent variables to consider in the linear part of the glm. For this function, only a binary variable is supported.
- `G.names`: The names of independent variables to consider in the tree part of the hybrid glm.
- `B`: The size of the bootstrap sample
- `args.rpart`: A list of options that control details of the `rpart` algorithm. Minimum number of observations in any terminal `<leaf>` node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
- `epsi`: A threshold value to check the convergence of the algorithm
- `iterMax`: The maximal number of iteration to consider
- `iterMin`: The minimum number of iteration to consider
- `family`: The binomial family.
- `LEVEL`: The level of the test
- `LB`: A binary indicator with values TRUE or FALSE indicating whether the loading is balanced or not in the parallel computing. It is useless on a windows platform.
- `args.parallel`: Parameters of the parallelization. See `mclapply` for more details.
- `verbose`: Logical; TRUE for printing progress during the computation (helpful for debugging)
Value

a list with six elements:
p.val_selected

the adjusted p-value of the selected tree

selected_model

a list with the fit of the selected pltr model fit glm, the selected tree tree
and the p-value of the selected tree without adjusting for multiple comparisons

p.value

fit_glm

the fitted pltr model under the null hypothesis if the test is not significant

Timediff

The execution time of the permutation test procedure

comp_p_values

The P-values of the competing trees

Badj

The number of samples used inside the procedure

Author(s)

Cyprien Mbogning

See Also

p.val.tree, best.tree.bootstrap

Examples

## Not run:

# load the data set

data(data_pltr)

# set the parameters

args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

# build a maximal tree

fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
family = family, iterMax = 5, iterMin = 3)

# select an test the selected tree by a permutation test procedure

args.parallel = list(numWorkers = 1, type = "PSOCK")

best_permute <- best.tree.permute(fit_pltr$tree, data_pltr, Y.name, X.names,
G.names, B = 10, args.rpart = args.rpart, epsi = 0.001, iterMax = 5,
iterMin = 3, family = family, LEVEL = 0.05, LB = FALSE,

## End(Not run) args.parallel = args.parallel)
The burn data frame has 154 rows and 17 columns.

Usage

data(burn)

Format

A data frame with 154 observations on the following 17 variables.

- **Obs**: Observation number
- **Z1**: Treatment: 0=routine bathing 1=Body cleansing
- **Z2**: Gender (0=male 1=female)
- **Z3**: Race: 0=nonwhite 1=white
- **Z4**: Percentage of total surface area burned
- **Z5**: Burn site indicator: head 1=yes, 0=no
- **Z6**: Burn site indicator: buttock 1=yes, 0=no
- **Z7**: Burn site indicator: trunk 1=yes, 0=no
- **Z8**: Burn site indicator: upper leg 1=yes, 0=no
- **Z9**: Burn site indicator: lower leg 1=yes, 0=no
- **Z10**: Burn site indicator: respiratory tract 1=yes, 0=no
- **Z11**: Type of burn: 1=chemical, 2=scald, 3=electric, 4=flame
- **T1**: Time to excision or on study time
- **D1**: Excision indicator: 1=yes 0=no
- **T2**: Time to prophylactic antibiotic treatment or on study time
- **D2**: Prophylactic antibiotic treatment: 1=yes 0=no
- **T3**: Time to streplococous aureaus infection or on study time
- **D3**: Streptococous aureaus infection: 1=yes 0=no

Source


Examples

data(burn)
## maybe str(burn) ;
Description

A data frame to test the functions of the package

Usage

data(data_pltr)

Format

A data frame with 3000 observations on the following 16 variables.

G1  a numeric vector
G2  a factor with levels 0 1
G3  a factor with levels 0 1
G4  a factor with levels 0 1
G5  a factor with levels 0 1
G6  a binary numeric vector
G7  a binary numeric vector
G8  a binary numeric vector
G9  a binary numeric vector
G10 a binary numeric vector
G11 a binary numeric vector
G12 a binary numeric vector
G13 a binary numeric vector
G14 a binary numeric vector
G15 a binary numeric vector
Y   a binary numeric vector

Details

The numeric variable G1 is considered as offset in the simulated PLTR model; the variables G2,...,G5 are used to simulate the tree part, while G6,...,G15 are noise variables.

Examples

data(data_pltr)
## maybe str(data_pltr) ...
nested.trees

**compute the nested trees**

**Description**

Compute a sequence of nested competing trees for the pruning step

**Usage**

```r
nested.trees(xtree, xdata, Y.name, X.names, MaxTreeSize = NULL,
             family = "binomial", verbose = TRUE)
```

**Arguments**

- `xtree`: a tree inheriting to the rpart method
- `xdata`: the dataset used to build the tree
- `Y.name`: the name of the dependent variable in the tree model
- `X.names`: the names of independent variables considered as offset in the tree model
- `MaxTreeSize`: The maximal size of the competing trees
- `family`: the glm family considered depending on the type of the dependent variable.
- `verbose`: Logical; TRUE for printing progress during the computation (helpful for debugging)

**Value**

a list with 4 elements:

- `leaves`: a list of leaves of the competing trees to consider for the optimal tree
- `null_deviance`: the deviance of the null model (linear part of the glm)
- `deviances`: a vector of deviances of the competing PLTR models
- `diff_deviances`: a vector of the deviance differences between the competing PLTR models and the null model

**Author(s)**

Cyprien Mbogning and Wilson Toussile

**Examples**

```r
## Not run:
## load the data set

data(data_pltr)
args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
```
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree

fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
                     family = family, iterMax = 5, iterMin = 3)

## compute the competing trees

nested_trees <- nested.trees(fit_pltr$tree, data_pltr, Y.name, X.names,
                              MaxTreeSize = 10, family = family)

## End(Not run)

---

**p.val.tree**

**Compute the p-value**

**Description**

Test whether the selected tree by either BIC, AIC or CV procedure is significantly associated to the dependent variable or not, while adjusting for a confounding effect.

**Usage**

```r
p.val.tree(xtree, xdata, Y.name, X.names, G.names, B = 10, args.rpart =
            list(minbucket = 40, maxdepth = 10, cp = 0), epsi = 0.001, iterMax = 5,
            iterMin = 3, family = "binomial", LB = FALSE,
            args.parallel = list(numWorkers = 1), index = 4, verbose = TRUE)
```

**Arguments**

- `xtree` the maximal tree obtained by the function pltr.glm
- `xdata` the data frame used to build xtree
- `Y.name` the name of the dependent variable
- `X.names` the names of independent confounding variables to consider in the linear part of the glm
- `G.names` the names of independent variables to consider in the tree part of the hybrid glm
- `B` the resampling size of the deviance difference
- `args.rpart` a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details
- `epsi` a threshold value to check the convergence of the algorithm
p.val.tree

iterMax      the maximal number of iteration to consider
iterMin      the minimum number of iteration to consider
family       the glm family considered depending on the type of the dependent variable.
LB           a binary indicator with values TRUE or FALSE indicating whether the loading
             are balanced or not in the parallel computing
args.parallel parameters of the parallelization. See mclapply for more details.
index        the size of the selected tree (by the functions best.tree.BIC.AIC or best.tree.CV)
             using one of the proposed criteria
verbose      Logical; TRUE for printing progress during the computation (helpful for debugging)

Value

A list of three elements:
p.value      The P-value of the selected tree
Timediff      The execution time of the test procedure
Badj          The number of samples used inside the the procedure

Author(s)

Cyprien Mbogning

References

Mbogning, C., Perdry, H., Toussile, W., Broet, P.: A novel tree-based procedure for deciphering the

See Also

best.tree.bootstrap, best.tree.permute

Examples

## Not run:
## load the data set
data(data_pltr)
## set the parameters
args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")
## build a maximal tree

```r
fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
                      family = family, iterMax = 5, iterMin = 3)
```

## pruned back the maximal tree by BIC or AIC criterion

```r
tree_select <- best.tree.BIC.AIC(xtree = fit_pltr$tree, data_pltr, Y.name,
                                 X.names, family = family)
```

## Compute the p-value of the selected tree by BIC

```r
args.parallel = list(numWorkers = 10, type = "PSOCK")
index = tree_select$best_index[[1]]
p_value <- p.val.tree(xtree = fit_pltr$tree, data_pltr, Y.name, X.names, G.names,
                      B = 100, args.rpart = args.rpart, epsi = 1e-3,
                      iterMax = 5, iterMin = 3, family = family, LB = FALSE,
                      args.parallel = args.parallel, index = index)
```

## End(Not run)

---

**pltr.glm**

*Partially tree-based regression model function*

### Description

The `pltr.glm` function is designed to fit an hybrid glm model with an additive tree part on a glm scale.

### Usage

```r
pltr.glm(data, Y.name, X.names, G.names, family = "binomial",
         args.rpart = list(cp = 0, minbucket = 20, maxdepth = 10),
         epsi = 0.001, iterMax = 5, iterMin = 3, verbose = TRUE)
```

### Arguments

- **data**: a data frame containing the variables in the model
- **Y.name**: the name of the dependent variable
- **X.names**: the names of independent variables to consider in the linear part of the glm
- **G.names**: the names of independent variables to consider in the tree part of the hybrid glm.
- **family**: the glm family considered depending on the type of the dependent variable.
- **args.rpart**: a list of options that control details of the rpart algorithm. `minbucket`: the minimum number of observations in any terminal <leaf> node; `cp`: complexity parameter (Any split that does not decrease the overall lack of fit by a factor of cp is not attempted); `maxdepth`: the maximum depth of any node of the final tree, with the root node counted as depth 0. ... See `rpart.control` for further details.
The `pltr.glm` function uses an iterative procedure to fit the linear part of the `glm` and the tree part. The tree obtained at the convergence of the procedure is a maximal tree which overfits the data. It's then mandatory to prune this tree by using one of the proposed criteria (BIC, AIC and CV).

**Value**

A list with four elements:

- `fit` the glm fitted on the confounding factors at the end of the iterative algorithm
- `tree` the maximal tree obtained at the end of the algorithm
- `nber_iter` the number of iterations used by the algorithm
- `Timediff` The execution time of the iterative procedure

**Note**

The tree obtained at the end of these iterative procedures usually overfits the data. It's therefore mandatory to use either `best.tree.BIC.AIC` or `best.tree.CV` to prune back the tree.

**Author(s)**

Cyprien Mbogning and Wilson Toussile

**References**


**See Also**

`rpart`
Examples

data(burn)

args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxcompete = 0,
                   maxsurrogate = 0)
family <- "binomial"
X.names = "Z2"
Y.name = "D2"
G.names = c("Z1", "Z3", "Z4", "Z5", "Z6", "Z7", "Z8", "Z9", "Z10", "Z11")

pltr.burn <- pltr.glm(burn, Y.name, X.names, G.names, args.rpart = args.rpart,
                       family = family, iterMax = 4, iterMin = 3, verbose = FALSE)

## Not run:
## load the data set

data(data_pltr)

## set the parameters

args.rpart <- list(minbucket = 40, maxdepth = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree

fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
                       family = family, iterMax = 4, iterMin = 3)

plot(fit_pltr$tree, main = "MAXIMAL TREE")
text(fit_pltr$tree, minlength = 0L, xpd = TRUE, cex = .6)

## End(Not run)

predict_bagg.pltr prediction on new features

Description

Prediction on new features using a set of bagging pltr models

Usage

predict_bagg.pltr(bag_pltr, Y.name, newdata, type = "response",
                  thresshold = seq(0, 1, by = 0.1))
predict_bagg.pltr

Arguments

  bag_pltr  the bagging result obtained with the function `bagging.pltr`
  Y.name the name of the binary dependent variable
  newdata a data frame in which to look for predictors and the dependant variable.
  type    the type of prediction required. `type = "response"` is the default; It gives the
           predicted probabilities. At this stage of the package, only this type is take into
           account. Other types such as "link" and "terms" are useless.
  threshold a vector of cutoff values for binary prediction. Could be helpfull for computing
           the AUC on the test sample.

Value

A list with 8 elements

FINAL_PRED_IND1
  A list of size the length of the threshold vector, containing the final prediction of
  each individual of the testing data by the bagging procedure using the majority
  rule (the modal prediction).

FINAL_PRED_IND2
  A list of size the length of the threshold vector, containing the final prediction
  of each individual of the testing data by the bagging procedure using the mean
  estimated probability.

PRED_ERROR1
  A vector of estimated errors of the Bagging procedure on the test sample for
  each threshold value using FINAL_PRED_IND1.

PRED_ERROR2
  A vector of estimated errors of the Bagging procedure on the test sample for
  each threshold value using FINAL_PRED_IND2.

CONF1
  A list of confusion matrix using FINAL_PRED_IND1

CONF2
  A list of confusion matrix using FINAL_PRED_IND2

PRED_ERRORS_PBP
  A list of size the length of the threshold vector. Each element representing the
  prediction error obtained via each predictor in the bagging sequence for each
  threshold value

PRED_ERROR_PBP
  A vector containing the mean of PRED_ERRORS_PBP for each threshold value

Author(s)

Cyprien Mbogning

References

  Mbogning, C., Perdry, H., Broet, P: A Bagged partially linear tree-based regression procedure for

See Also

  `bagging.pltr`, `predict.glm`
Examples

```r
## Not run:
## load the data set
data(burn)

## set the parameters
args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxsurrogate = 0)
family <- "binomial"
Y.name <- "D2"
X.names <- "Z2"
G.names <- c("Z1", "Z3", "Z4", "Z5", "Z6", "Z7", "Z8", "Z9", "Z10", "Z11")
args.parallel = list(numWorkers = 1)

## Bagging a set of basic unpruned pltr predictors
Bag.burn <- bagging.pltr(burn, Y.name, X.names, G.names, family,
                        args.rpart, epsi = 0.01, iterMax = 4, iterMin = 3,
                        Bag = 20, verbose = FALSE, doprune = FALSE)

## Use the bagging procedure to predict new features
# ?predict_bagg.pltr
Pred_Bag.burn <- predict_bagg.pltr(Bag.burn, Y.name, newdata = burn,
                        type = "response", thresshold = seq(0, 1, by = 0.1))

## The confusion matrix for each thresshold value using the majority vote
Pred_Bag.burn$CONF1

## End(Not run)
```

predict_pltr

### Description

prediction on new features using a pltr tree and the name of the confounding variable

### Usage

```r
predict_pltr(xtree, xdata, Y.name, X.names, newdata, type = "response",
             family = "binomial", thresshold = seq(0.1, 0.9, by = 0.1))
```

### Arguments

- **xtree**: a tree obtained with the pltr procedure
Description

fit the PLTR model for a given tree. The tree is coerced into dummy covariates.

Usage

tree2glm(xtree, xdata, Y.name, X.names, family = "binomial")
Arguments

- **xtree**: a tree inherits from the rpart method
- **xdata**: a data frame containing the variables in the model
- **Y.name**: the name of the dependent variable
- **X.names**: the names of independent variables to consider in the linear part of the glm
- **family**: the glm family considered depending on the type of the dependent variable.

Value

the pltr fitted model (fit)

Author(s)

Cyprien Mbogning and Wilson Toussile

Examples

```r
## Not run:
# load the data set
data(data_pltr)

## set the parameters
args.rpart <- list(minbucket = 40, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree
fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
family = family, iterMax = 5, iterMin = 3)

## Coerce a tree into a glm model using the confounding factor
fit_glm <- tree2glm(fit_pltr$tree, data_pltr, Y.name, X.names,
family = family)

summary(fit_glm)

## End(Not run)
```
tree2indicators

From a tree to indicators (or dummy variables)

Description

Coerces a given tree structure to binary covariates.

Usage

tree2indicators(fit)

Arguments

fit a tree structure inheriting to the rpart method

Value

a list of indicators

Author(s)

Cyprien Mbogning and Wilson Toussile

Examples

## Not run:
## load the data set
data(data_pltr)

## set the parameters
args.rpart <- list(minbucket = 40, xval = 10, cp = 0)
family <- "binomial"
Y.name <- "Y"
X.names <- "G1"
G.names <- paste("G", 2:15, sep="")

## build a maximal tree
fit_pltr <- pltr.glm(data_pltr, Y.name, X.names, G.names, args.rpart = args.rpart,
                      family = family, iterMax = 5, iterMin = 3)

## Compute a list of indicator from the leaves of the tree fitted tree
tree2indicators(fit_pltr$tree)

## End(Not run)
VIMPBAG  

score of importance for variables

Description

Several variable importance scores are computed: the deviance importance score (DIS), the permutation importance score (PIS), the depth deviance importance score (DDIS), the minimal depth importance score (MinDepth) and the occurrence score (OCCUR).

Usage

VIMPBAG(BAGGRES, data, Y.name)

Arguments

- **BAGGRES**: The output of the bagging procedure (bagging.pltr)
- **data**: The learning dataframe used within the bagging procedure
- **Y.name**: The name of the binary dependant variable used in the bagging procedure

Details

Several choices for variable selection using the bagging procedure are proposed. A discussion about the scores of importance PIS, DIS, and DDIS is available in Mbogning et al. 2015

Value

A list with 9 elements

- **PIS**: A list of length the length of the threshold value used in the bagging procedure, containing the permutation importance score displayed in decreasing order for each threshold value
- **StdPIS**: The standard error of the PIS
- **OCCUR**: The occurrence number for each variable in the bagging sequence displayed in decreasing order
- **DIS**: The deviance importance score displayed in decreasing order
- **DDIS**: The depth deviance importance score displayed in decreasing order
- **MinDepth**: The minimal depth score for each variable, displayed in increasing order
- **dimtrees**: A vector containing the dimensions of trees within the bagging sequence
- **EOOB**: A vector containing the OOB error of the bagging procedure for each threshold value
- **Bagfinal**: The number of Bagging iterations used

Author(s)

Cyprien Mbogning
References

See Also

bagging.pltr

Examples

```r
## Not run:
## load the data set
data(burn)

## set the parameters
args.rpart <- list(minbucket = 10, maxdepth = 4, cp = 0, maxsurrogate = 0)
family <- "binomial"
Y.name <- "D2"
X.names <- "Z2"
G.names <- c("Z1", "Z3", "Z4", "Z5", "Z6", "Z7", "Z8", "Z9", "Z10", "Z11")
args.parallel = list(numWorkers = 1)

## Bagging a set of basic unprunned pltr predictors
Bag.burn <- bagging.pltr(burn, Y.name, X.names, G.names, family, 
args.rpart, epsi = 0.01, iterMax = 4, iterMin = 3, 
Bag = 20, verbose = FALSE, doprune = FALSE)

## Several importance scores for variables, using the bagging procedure
Var_Imp_BAG.burn <- VIMPBAG(Bag.burn, burn, Y.name)

## Importance score using the permutaion method for each thresshold value
Var_Imp_BAG.burn$PIS

## Importance score using the deviance criterion
Var_Imp_BAG.burn$DIS

## End(Not run)
```
Index

* Machine Learning
  - bagging.pltr, 6
  - best.tree.CV, 13
  - pltr.glm, 22
  - predict_bagg.pltr, 24
  - VIMPBAG, 30

* datasets
  - burn, 17
  - data_pltr, 18

* documentation
  - bag.aucoob, 5
  - bagging.pltr, 6
  - best.tree.BIC.AIC, 9
  - best.tree.bootstrap, 11
  - best.tree.CV, 13
  - best.tree.permute, 15
  - nested.trees, 19
  - p.val.tree, 20
  - pltr.glm, 22
  - predict_bagg.pltr, 24
  - predict_pltr, 26
  - tree2glm, 27
  - tree2indicators, 29
  - VIMPBAG, 30

* package
  - GPLTR-package, 2

* test
  - best.tree.bootstrap, 11
  - best.tree.permute, 15
  - p.val.tree, 20

* tree
  - best.tree.BIC.AIC, 9
  - pltr.glm, 22

* variable selection
  - VIMPBAG, 30

bag.aucoob, 5
bagging.pltr, 5, 6, 25, 31
best.tree.BIC.AIC, 9, 14, 21, 23
best.tree.bootstrap, 11, 16, 21
best.tree.CV, 10, 13, 21, 23
best.tree.permute, 15, 21
burn, 17
data_pltr, 18
GPLTR (GPLTR-package), 2
GPLTR-package, 2
mclapply, 7, 11, 15, 21
nested.trees, 19
p.val.tree, 12, 16, 20
pltr.glm, 10, 14, 22, 27
predict.glm, 25, 27
predict_bagg.pltr, 8, 24
predict_pltr, 26
rpart, 23
rpart.control, 6, 11, 13, 15, 20, 22
tree2glm, 27
tree2indicators, 29
VIMPBAG, 30