Package 'hhcartr'

July 2, 2021

Type Package

Title HHCART(G) - A Reflected Feature Space for CART

Version 1.0.0

Author Phil Davies [aut, cre]

Maintainer Phil Davies <phil.davies@pg.canterbury.ac.nz>

Description An implementation of the HHCART-G algorithm as described in the paper - Wickramarachchi C, Robertson B, Reale M, Price C, Brown J (2019). 'A reflected feature space for CART.' Australian & New Zealand Journal of Statistics, 61, 380–391. <doi:10.1111/anzs.12275>.

Depends R (>= 3.6.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports checkmate, hash, Matrix, DiagrammeR, ggplot2, DiagrammeRsvg, captioner, rsvg, bookdown,

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-07-02 07:00:08 UTC

R topics documented:

bagging_predict *bagging_predict Make predictions for each test dataset row against each tree.*

Description

This internal function is used to perform some basic checks on the input dataset. The first check to fail will stop the model from being instantiated. The following checks are performed:

Usage

```
bagging_predict(mytrees, testx, useIdentity, classify, objectid)
```

Arguments

mytrees	A list of all trees.
testx	The test dataset, the target variables are in the last column.
useIdentity	Whether the training data has been transformed with the householder transform.
classify	Default is TRUE. Set TRUE for a classification problem and FALSE for a regression problem.
objectid	List of objectids (tree nodes) that will be collapsed in the current tree.

Value

A matrix of all predictions made from all trees and another matrix with a margin for each tree.

balance *Balance Scale Dataset.*

Description

This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance x left-weight) and (right-distance x right-weight). If they are equal, it is balanced.

Usage

```
data(balance)
```

Format

A data frame with 625 rows and 4 variables:

Left-Weight Left-Weight, one of 1, 2, 3, 4, or 5
Left-Distance Left-Distance, one of 1, 2, 3, 4, or 5
Right-Weight Right-Weight, one of 1, 2, 3, 4, or 5
Right-Distance Right-Distance, one of 1, 2, 3, 4, or 5
Class Name Class Name: one of L, B or R)

Source

https://archive.ics.uci.edu/ml/datasets/Balance+Scale

---

best_split_ *best_split_ finds the best feature column to split on.*

Description

This internal function is used to find the feature column that will offer the best split based on using the Gini index or gini hyperplane index.

Usage

```
best_split_(
  X,
  y,
  most_freq_class,
  split_original,
  n_classes,
  max_features,
  depth
)
```

Arguments

X	feature variables to search for the best split.
y	target variable.
most_freq_class	the most frequent class in the target variable.
split_original	boolean to indicate whether to split on original data or reflected data.
n_classes	number of classes in the y column
max_features	the maximum number of features to use when splitting a node
depth	the depth of the current tree.

Value

a list of the following variables (best_idx, best_thr, best_gini)

bupa *Liver Disorders Dataset.*

Description

The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the dataset constitutes the record of a single male individual.

Usage

```
data(bupa)
```

Format

A data frame with 345 rows and 7 variables:

mcv mean corpuscular volume

alkphos alkaline phosphotase

sgpt alanine aminotransferase

sgot aspartate aminotransferase

gammagt gamma-glutamyl transpeptidase

drinks number of half-pint equivalents of alcoholic beverages drunk per day

selector field created by the BUPA researchers to split the data into train/test sets

Source

https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

---

calculate_margin_for_tree
calculate_margin_for_tree() - Calculate the margin for a tree.

Description

This internal function is called from bagging_predict to calculate the margin on the current tree using the test set.

Usage

```
calculate_margin_for_tree(preds, actuals, count_classified_correct, total)
```

Arguments

`preds`	Predictions on the test set.
`actuals`	The actuals classes ie. the target variables.
`count_classified_correct`	The number of correctly classified predictions.
`total`	The number of rows in the test set.

Value

returns the margin for the current tree.

`cancer` *Wisconsin Breast Cancer.*

Description

This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

Usage

```
data(cancer)
```

Format

A data frame with 683 rows and 10 variables:

Sample code number id number

Clump Thickness 1-10

Uniformity of Cell Size 1-10

Uniformity of Cell Shape 1-10

Marginal Adhesion 1-10

Single Epithelial Cell Size 1-10

Bare Nuclei 1-10

Bland Chromatin 1-10

Normal Nucleoli 1-10

Mitoses 1-10

Class 2 for benign, 4 for malignant

Details

Attributes 2 through 10 have been used to represent instances. Each instance has one of 2 possible classes: benign or malignant. Original dataset had 699 samples, 16 were removed as these contained NA's. Class distribution: Benign: 458 (65.5

Source

https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)

`compute_min_node_impurity` *compute_min_node_impurity.*

Description

This function instantiates a HHDecisionTree model, it is used to induce classification or regression trees depending upon the value of the response parameter. It supports the parameters listed below.

Usage

```
compute_min_node_impurity(X, y, response, seed, useIdentity = FALSE)
```

Arguments

`X`	The feature variables we will use to train the HHDecisionTree model on.
`y`	The target variable.
`response`	The response parameter is used to specify what type of model to build, either 'classify' for a classification tree model or 'regressor' for a regression tree model. The default is 'classify'.
`seed`	The random seed to be used to prime the random number generator.
`useIdentity`	The useIdentity parameter when set TRUE will result in hhcartr using the original training data to find the optimal splits rather than using the reflected data. The default value is FALSE.

Value

Returns the "best" value of min_node_impurity.

`displayTree` *displayTree Display one selected decision tree created from DOT statements.*

Description

This function displayTree() generates DOT statements for the selected decision tree in the current model. The DOT statements are written to a temporary file in the tmp directory. Function grViz() from the DiagrammeR package is then called to visualize the graph. displayTree() will check to make sure package DiagrammeR is installed and loaded before attempting generation of DOT statements.

Usage

```
displayTree(ntree = 1, rpart_ = NA)
```

Arguments

ntree	The number of the tree the user wishes to display.
rpart_	If specified, it is the rpart() equivalent tree in hhcartr format.

Value

nothing.

Examples

```
# source: /man/examples/displayTree.R

# basic usage of displayTree(n)

# Note: need to have something to display first.
X <- iris[,1:4]
y <- iris[,5]
clf = HHDecisionTree(n_folds=1,
                     n_trees=1,
                     pruning=FALSE,
                     min_node_impurity=0.0)
# train our model.
vv <- clf$fit(X, y)
# display the resulting tree.
displayTree(1)
```

---

dispnode *dispnode generates DOT statements for all trees in current model.*

Description

This internal function generates DOT statements for all trees in the current model. It is called by function navigate_hash.

Usage

```
dispnode(nn, level, pid)
```

Arguments

nn	current node of the current tree
level	current depth in the current tree
pid	the parent-id of the current node in the current tree

Value

nothing

`getCoefficients` *getCoefficients Get the coefficents from the house holder matrix for the selected node.*

Description

This function getCoefficients() returns the coefficients from the appropriate column of the house holder matrix for the selected internal node or for all internal nodes if no parameter was specified.

Usage

```
getCoefficients(n_node, fold)
```

Arguments

`n_node`	The internal node number the user wants coefficients for. If no node number is supplied then coefficients for all internal nodes of the current tree are returned.
`fold`	The fold of the corresponding tree we want coefficients for.

Value

A dataframe containing the requested information. NA is returned if the requested node was either not found or was a leaf node.

`glass` *Glass Identification Dataset.*

Description

The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence... if it is correctly identified!

Usage

```
data(glass)
```

Format

A data frame with 214 rows and 9 variables:

Id number id number

RI refractive index

Na Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)

Mg Magnesium

Al Aluminium

Si Silicon

K Potassium

Ca Calcium

Ba Barium

Fe Iron

Type of Glass (class attribute) – 1 building_windows_float_processed –2 building_windows_non_float_processed – 3 vehicle_windows_float_processed –4 vehicle_windows_non_float_processed (none in this database) – 5 containers – 6 tableware – 7 headlamps

Details

From USA Forensic Science Service; 6 types of glass; defined in terms of their oxide content (i.e. Na, Fe, K, etc).

Source

https://archive.ics.uci.edu/ml/datasets/Glass+Identification

`grow_tree_` *grow_tree_*

Description

This internal function is used to grow the decision tree, it is called recursively until the stopping criteria are met.

Usage

```
grow_tree_(
  node,
  X,
  y,
  n_min,
  min_node_impurity,
  useIdentity,
  classify = TRUE,
```

```
  n_features,
  n_classes,
  max_features,
  depth = 0
)
```

Arguments

node	The current node to be tested for a split.
X	The feature variables of the current node being processed.
y	The target variable for the feature variables.
n_min	The n_min parameter is used to terminate node splitting when a minimum number of samples at that node has been reached. The default value is 2.
min_node_impurity	Splitting a node will stop when the impurity of a node is less than min_node_impurity. The node impurity is calculated using the hyperplane Gini index. The default value is 0.2.
useIdentity	The useIdentity parameter when set TRUE will result in hhcartr using the original training data to find the optimal splits rather than using the reflected data. The default value is FALSE.
classify	The classify parameter when set TRUE will result in hhcartr performing a classification, when set FALSE will perform a regression.
n_features	The number of feature variables.
n_classes	The number of classes in the target variable.
max_features	The maximum number of features to consider in the current split.
depth	This parameter is not used as yet.

Details

The following parameters are supported:

Value

Returns the latest node of type NNode.

---

heart *Heart Disease Dataset.*

Description

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).

Usage

```
data(heart)
```

Format

A data frame with 303 rows and 13 variables:

age age in years

sex sex (1 = male; 0 = female)

cp cp: chest pain type

trestbps trestbps: resting blood pressure (in mm Hg on admission to the hospital)

chol chol: serum cholestoral in mg/dl

fbs fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

restecg restecg: resting electrocardiographic results

thalach thalach: maximum heart rate achieved

exang exang: exercise induced angina (1 = yes; 0 = no)

oldpeak oldpeak = ST depression induced by exercise relative to rest

slope slope: the slope of the peak exercise ST segment

ca ca: number of major vessels (0-3) colored by flourosopy

thal thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

num num: diagnosis of heart disease (angiographic disease status)

Source

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

---

hhcartr_regressor_find_better_split

hhcartr_regressor_find_better_split finds the best feature to split on.

Description

This internal function is used by the regressor model to find the feature column that will offer the best split based on sum of squares.

Usage

```
hhcartr_regressor_find_better_split(X, y, max_features)
```

Arguments

X	feature variables to search for the best split.
y	target variable.
max_features	the maximum number of features to use when splitting a node

Value

a list of the following variables (var_idx_, split_, score_)

`hhcart_reflect_feature_space_g` *hhcart_reflect_feature_space_g*

Description

This function is in an internal only function. It is used to reflect the feature data if the data is found to be suitable, otherwise node splitting will be done using the original data and using axis-parallel splits.

Usage

```
hhcart_reflect_feature_space_g(
  X,
  y,
  useIdentity = FALSE,
  max_features = "None",
  n_features = n_features,
  n_classes = n_classes,
  classify = TRUE,
  depth = depth,
  colx = 1
)
```

Arguments

`X`	The feature variables of the current node being processed.
`y`	The target variable for the corresponding feature variables.
`useIdentity`	A flag, if FALSE the node split will be done using an oblique split after transforming the training data using a householder transform. If TRUE the 'transform' will be the identity and the split will be done on the original data.
`max_features`	The maximum number of features to consider in the current split.
`n_features`	The number of feature variables.
`n_classes`	The number of classes in the target variable.
`classify`	A flag, if TRUE this is a classification problem, if FALSE this is a regression problem.
`depth`	The current tree depth.
`colx`	The eigenvector column to use.

Details

The following parameters are supported:

X_matrix, X, y, most_freq_class, n_classes, max_features, n_features, colx

Value

Returns a list containing idx, thr, X_house, householder_matrix, using_householder if split was for a classification problem and reflected data used, otherwise the results from split_using_original_data are returned if useIdentity was TRUE. For a regression problem a list containing the following is returned: idx, thr, as.matrix(X), NULL, FALSE.

`hhcart_run_classifier` *hhcart_run_classifier - This function calls grow_tree_() to induce a decision tree.*

Description

hhcart_run_classifier - This function calls grow_tree_() to induce a decision tree.

Usage

```
hhcart_run_classifier(
  train_set,
  sample_size,
  j,
  n_min,
  min_node_impurity,
  sampleWithReplacement,
  useIdentity,
  classify,
  n_features,
  n_classes,
  max_features = NA
)
```

Arguments

`train_set`	The training data.
`sample_size`	The sample size parameter is used to determine how much of the training dataset is actually used during training. A value of 1.0 allows all of the current training dataset to be used for training. A value of less than one will mean that proportion of the training dataset will be selected at random and then used for training. The value of parameter sampleWithReplacement will determine if the random sampling of the training dataset is performed using replacement or not. The default value is 1.0.
`j`	Not used, kept for compatibility.

`n_min` The n_min parameter is used to stop splitting a node when a minimum number of samples at that node has been reached. The default value is 2.

`min_node_impurity` The min node impurity parameter is used to stop splitting a node if the node impurity at that node is less than this value. The node impurity is calculated using the hyperplane Gini index. The default value is 0.2.

`sampleWithReplacement` The sampleWithReplacement parameter is used in conjunction with the sample size parameter. The sampleWithReplacement parameter will determine if sampling from the training dataset is done with or without replacement. The default value is FALSE.

`useIdentity` The useIdentity parameter when set TRUE will result in hhcartr using the original training data to find the optimal splits rather than using the reflected data. The default value is FALSE.

`classify` The classify parameter when set TRUE indicates that the data is for building a classification model. A value of FALSE and a regression model will be induced.

`n_features` The number of features in the training data.

`n_classes` The number of classes in the training data.

`max_features` The max features parameter determines the number of features to consider when looking for the best split, and can take one of the values listed below. The default value is "sqrt".

Value

Returns an induced decision tree.

`hhcart_verify_input_data` *hhcart_verify_input_data verify the input data.*

Description

This internal function is used to perform some basic checks on the input dataset. The first check to fail will stop the model from being instantiated. The following checks are performed: - there must be no NA's in the training dataset. - all columns in the training datasets must contain numeric data only. - max_features must not be greater than n_features. - if its a classification problem the target variable must be categorical. - if its a regression problem the target variable must not be categorical.

Usage

```
hhcart_verify_input_data(
  X,
  y,
  classify = TRUE,
  max_features_ = 0,
```

```
  n_features_ = 0,
  rforest = FALSE
)
```

Arguments

X	The training dataset.
y	target variable column.
classify	Default is TRUE. Set TRUE for a classification problem and FALSE for a regression problem.
max_features_	Default is 0. The maximum number of features to search for an optimal split.
n_features_	Default is 0. The total number of feature columns in the training dataset.
rforest	Default is FALSE. Indicates whether this function is being called from HHDecisionTree or HHRandomForest.

Value

Nothing if all checks are passed, otherwise the function stops.

HHDecisionTree *HHDecisionTree.*

Description

This function instantiates a HHDecisionTree model, it is used to induce classification or regression trees depending upon the value of the response parameter. It supports the parameters listed below.

Usage

```
HHDecisionTree(
  response = "classify",
  n_min = 2,
  min_node_impurity = 0.2,
  n_trees = 1,
  n_folds = 5,
  testSize = 0.2,
  useIdentity = FALSE,
  pruning = FALSE,
  dataDescription = "Unknown",
  control = mni.control(n_folds = 5),
 prune_control = prune.control(prune_type = "all", prune_stochastic_max_nodes = 14,
    prune_stochastic_max_depth = 20, prune_stochastic_samples = 3000),
  show_progress = FALSE,
  seed = NA
)
```

Arguments

response	The response parameter is used to specify what type of model to build, either 'classify' for a classification tree model or 'regressor' for a regression tree model. The default is 'classify'.
n_min	The n_min parameter is used to stop splitting a node when a minimum number of samples at that node has been reached. The default value is 2.
min_node_impurity	The min node impurity parameter is used to stop splitting a node if the node impurity at that node is less than this value. The node impurity is calculated using the hyperplane Gini index. The default value is 0.2.
n_trees	The n_trees parameter is used to specify the number of trees to use(grow) per fold or trial. The default value is 1.
n_folds	The n_folds parameter is used to specify the number of folds to use i.e. split the input data into n_folds equal amounts, for n_folds times, use one portion of the input data as a test dataset, and the remaining n_folds-1 portions as the training dataset. The model is then trained using these training and test datasets, once training is complete the next fold or portion of the input dataset is treated as the test dataset and the remainder the training dataset, the model is then trained again. This process is repeated until all portions or folds of the input dataset have been used as a test dataset. When n_folds=1 the testSize parameter determines the size of the test dataset. The default value is 5.
testSize	The testSize parameter determines how much of the input dataset is to be used as the test dataset. The remainder is used as the training dataset. This parameter is only used when the parameter n_folds=1. For values of n_folds greater than one, the computed fold size will govern the test dataset size used (see the n_folds parameter for more details). The default value is 0.2.
useIdentity	The useIdentity parameter when set TRUE will result in hhcartr using the original training data to find the optimal splits rather than using the reflected data. The default value is FALSE.
pruning	The pruning parameter when set TRUE will result in tree pruning after all trees are induced. The default value is FALSE.
dataDescription	The dataDescription parameter is a short description used to describe the dataset being modelled. It is used is output displays and plots as documentation. The default value is "Unknown".
control	The control parameter is used to specify parameters for the mni.control function. See documentation for mni.control for supported parameters.
prune_control	The prune_control parameter is used to specify parameters for the prune.control function. This parameter is only used when 'pruning = TRUE'. See documentation for prune.control for supported parameters.
show_progress	The show_progress parameter when set TRUE will cause progress messages to be displayed as trees are induced. A value of FALSE will result in no progress messages being displayed. The default value is TRUE.
seed	Specify a seed to seed the RNG. Acceptable values are 1-9999. If no value is specified a random integer in the range 1-9999 is used.

Value

Returns pkg.env$folds_trees, a list of all trees induced during training.

`HHDecisionTreeCore` *HHDecisionTreeCore Common function for all hhcartr model functions.*

Description

This function internal function provides a common interface for all hhcartr model function. At the time of writing these are HHDecisionTreeClassifier and HHDecisionTreeRegressor. The following parameters are supported (they are not necessarily all common to the classifier and regressor models - look at documentation for each model).

Usage

```
HHDecisionTreeCore(
  response = "classify",
  n_min = 2,
  min_node_impurity = 0.2,
  n_trees = 1,
  n_folds = 5,
  sample_size = 1,
  testSize = 0.2,
  sampleWithReplacement = FALSE,
  useIdentity = FALSE,
  dataDescription = "Unknown",
  max_features = "None",
  pruning = FALSE,
  parallelize = FALSE,
  number_cpus = 1,
  show_progress = FALSE,
  seed = seed,
  control = control,
  prune_control = prune_control,
  debug_msgs = FALSE
)
```

Arguments

`response`	The response parameter is used to specify what type of model to build, either 'classify' for a classification tree model or 'regressor' for a regression tree model. The default is 'classify'.
`n_min`	The n min parameter is used to stop splitting a node when a minimum number of samples at that node has been reached. The default value is 2.

`min_node_impurity`
: The min node impurity parameter is used to stop splitting a node if the node impurity at that node is less than this value. The node impurity is calculated using the hyperplane Gini index. The default value is 0.2.

`n_trees`
: The n trees parameter is used to specify the number of trees to use(grow) per fold or trial. The default value is 1.

`n_folds`
: The n folds parameter is used to specify the number of folds to use i.e. split the input data into n folds equal amounts, for n folds times, use one portion of the input data as a test dataset, and the remaining n folds-1 portions as the training dataset. The model is then trained using these training and test datasets, once training complete the next fold or portion of the input dataset is treated as the test dataset and the remainder the training dataset, the model is then trained again. This process is repeated until all portions or folds of the input dataset have been used as a test dataset. The default value is 5.

`sample_size`
: The sample size parameter is used to determine how much of the training dataset is actually used during training. A value of 1.0 allows all of the current training dataset to be used for training. A value of less than one will mean that proportion of the training dataset will be selected at random and then used for training. The value of parameter sampleWithReplacement will determine if the random sampling of the training dataset is performed using replacement or not. The default value is 1.0.

`testSize`
: The testSize parameter determines how much of the input dataset is to be used as the test dataset. The remainder is used as the training dataset. This parameter is only used when the parameter n_folds=1. For values of n_folds greater than one, the computed fold size will govern the test dataset size used (see the n_folds parameter for more details). The default value is 0.2.

`sampleWithReplacement`
: The sampleWithReplacement parameter is used in conjunction with the sample size parameter. The sampleWithReplacement parameter will determine if sampling from the training dataset is done with or without replacement. The default value is FALSE.

`useIdentity`
: The useIdentity parameter when set TRUE will result in hhcartr using the original training data to find the optimal splits rather than using the reflected data. The default value is FALSE.

`dataDescription`
: The dataDescription parameter is a short description used to describe the dataset being modelled. It is used is output displays and plots as documentation. The default value is "Unknown".

`max_features`
: The max features parameter determines the number of features to consider when looking for the best split, and can take one of the values listed below. The default value is "sqrt".

`pruning`
: The pruning parameter when set TRUE specifies that the induced tree is to be pruned after induction. The default value is FALSE.

`parallelize`
: The parallelize parameter when set TRUE will allow selected loops to be run in parallel. (This functionality has yet to be fully tested). The default value is FALSE.

number_cpus	The number of available CPU's to use when parameter parallelize is set to TRUE. The maximum number of CPU's to be used will be the number of physical CPU's available (as returned via the detectCores() function of the parallel package) minus one. The default value is 1.
show_progress	The show_progress parameter when set TRUE will cause progress messages to be displayed as trees are induced. A value of FALSE will result in no progress messages being displayed. The default value is FALSE.
seed	Specify a seed to seed the RNG. Acceptable values are 1-9999. If no value is specified a random integer in the range 1-9999 is used.
control	Default value mni.control(n_folds = 5). The control parameter is used to specify parameters for the mni.control function. See documentation for mni.control for supported parameters.
prune_control	Default value prune.control(prune_type = "all", prune_fatbears_max_nodes = 10, prune_fatbears_iterations = 1000) The prune_control parameter is used to specify parameters for the prune.control function. This parameter is only used when 'pruning = TRUE'. See documentation for prune.control for supported parameters.
debug_msgs	Not fully implemented yet but will turn on debug messages.
classify	The classify parameter when set TRUE indicates that the data is for building a classification model. A value of FALSE and a regression model will be induced.

Value

a list of the trees induced during training, these are saved in global enviornment variable pkg.env$folds_trees.

---

housing *Boston Housing dataset.*

Description

Concerns housing values in suburbs of Boston.

Usage

```
data(housing)
```

Format

A data frame with 506 rows and 13 variables:

CRIM per capita crime rate by town

ZN proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS proportion of non-retail business acres per town

CHAS Charles River dummy variable (= 1 if tract bounds river, 0 otherwise)

NOX nitric oxides concentration (parts per 10 million)

RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to 1940

DIS weighted distances to five Boston employment centres

RAD index of accessibility to radial highways

TAX full-value property-tax rate per $10,000

PTRATIO pupil-teacher ratio by town

B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT percentage lower status of the population

MEDV Median value of owner-occupied homes in $1000s

Details

For regression models, build models to predict the MEDV field. For classification models hhcartr provides a binary target variable calculated using MEDV > 20 which can then be used to build a classification model.

Source

http://archive.ics.uci.edu/ml/machine-learning-databases/housing/

`invoke_model` *invoke_model.*

Description

This function instantiates a HHDecisionTree model, it is used to induce classification or regression trees depending upon the value of the response parameter. It supports the parameters listed below. It is used to find the optimum value for min_node_impurity.

Usage

```
invoke_model(
  X,
  y,
  try_this_min_imp,
  iter,
  response,
  mni_n_folds,
  mni_n_trees,
  useIdentity,
  seed
)
```

Arguments

X	The feature variables we will use to train the HHDecisionTree model on.
y	The target variable.
try_this_min_imp	The minimum node impurity we will try on this invocation of the HHDecisionTree model.
iter	The number of times function invoke_model has been invoked, used to increment a number that is passed to the function set.seed().
response	The response parameter is used to specify what type of model to build, either 'classify' for a classification tree model or 'regressor' for a regression tree model. The default is 'classify'.
mni_n_folds	The number of folds to pass to mni.control to be used to instantiate a HHDecisionTree model when searching for an optimum value of min_node_impurity.
mni_n_trees	The number of trees to pass to mni.control to be used to instantiate a HHDecisionTree model when searching for an optimum value of min_node_impurity.
useIdentity	The useIdentity parameter when set TRUE will result in hhcartr using the original training data to find the optimal splits rather than using the reflected data. The default value is FALSE.
seed	The seed parameter is used to provide a seed value for the R random number generator to ensure repeatable experiments.

Value

Returns statistics on the current run of the HHDecisionTree model.

---

landsat *Statlog (Landsat Satellite) Dataset.*

Description

The database consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values. In the sample database, the class of a pixel is coded as a number. The Landsat satellite data is one of the many sources of information available for a scene. The interpretation of a scene by integrating spatial data of diverse types and resolutions including multi-spectral and radar data, maps indicating topography, land use etc. is expected to assume significant importance with the onset of an era characterised by integrative approaches to remote sensing (for example, NASA's Earth Observing System commencing this decade). Existing statistical methods are ill-equipped for handling such diverse data types. Note that this is not true for Landsat MSS data considered in isolation (as in this sample database). This data satisfies the important requirements of being numerical and at a single resolution, and standard maximum-likelihood classification performs very well. Consequently, for this data, it should be interesting to compare the performance of other methods against the statistical approach. One frame of Landsat MSS imagery consists of four digital images of the same scene in different

spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red. Each pixel is a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80m x 80m. Each image contains 2340 x 3380 such pixels.

Usage

```
data(landsat)
```

Format

A data frame with 6435 rows and 36 variables:

cols1-36 The attributes are numerical, in the range 0 to 255.

class Number Class 1 red soil, 2 cotton crop, 3 grey soil, 4 damp grey soil, 5 soil with vegetation stubble, 6 mixture class (all types present), 7 very damp grey soil

Details

The database is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighbourhood of pixels completely contained within the 82x100 sub-area. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighbourhood and a number indicating the classification label of the central pixel.

Source

https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)

---

letters *Letter Recognition Dataset.*

Description

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We typically train on the first 16000 items and then use the resulting model to predict the letter category for the remaining 4000. See the article cited above for more details.

Usage

```
data(letters)
```

Format

A data frame with 20,000 rows and 16 variables:

x-box horizontal position of box (integer)

y-box vertical position of box (integer)

width width of box (integer)

high height of box (integer)

onpix total # on pixels (integer)

x-bar mean x of on pixels in box (integer)

y-bar mean y of on pixels in box (integer)

x2bar mean x variance (integer)

y2bar mean y variance (integer)

xybar mean x y correlation (integer)

x2ybr mean of x * x * y (integer)

xy2br mean of x * y * y (integer)

x-ege mean edge count left to right (integer)

xegvy correlation of x-ege with y (integer)

y-ege mean edge count bottom to top (integer)

yegvx correlation of y-ege with x (integer)

lettr (class attribute) capital letter (26 values from A to Z)

Source

https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

`make_predictions` *make_predictions Make predictions on the test dataset.*

Description

This internal function is used to make predictions on the test dataset against all induced trees.

Usage

```
make_predictions(list_trees, test, useIdentity, classify, objectid)
```

Arguments

list_trees	List of all induced trees.
test	The test dataset.
useIdentity	Whether the training data has been transformed with the householder transform.
classify	Default is TRUE. Set TRUE for a classification problem and FALSE for a regression problem.
objectid	A list of node numbers that will be 'pruned' ie. when making predictions if the tree node matches a node in objectid the tree node will be used to make the prediction rather than traversing any underlying nodes.

Value

Accuracy, margin and predictions.

---

mni.control *mni.control*

Description

This internal function is used to validate the parameters specified on the control parameter.

Usage

```
mni.control(
  mni_trials = 1,
  mni_n_folds = 10,
  mni_n_trees = 1,
  mni_size = 0.01,
  mni_start = 0.05,
  mni_numvals = 50,
  ...
)
```

Arguments

mni_trials	The number of times the process is repeated i.e. how many times the n-fold cross-validation process is repeated. The resulting value of min_node_impurity is the mean of mni_trials trials. The default value is 1.
mni_n_folds	The number of folds to use when evaluating values of min_node_impurity. Any integer value in the range 1 to the number of observations in the training dataset is accepted. The default value is 10.
mni_n_trees	The number of trees to use when evaluating values of min_node_impurity. At the time of writing the only allowable value is 1. The default value is 1.

mni_size	After a value of min_node_impurity is tried, the next value is calculated by adding mni_size to the previous value. A value in the range 0.001 to 0.10 is accepted. The default value is 0.01.
mni_start	The first value of min_node_impurity to be evaluated. A value in the range 0.001 to 1.0 is accepted. The default value is 0.05.
mni_numvals	The number of min_node_impurity values to try while attempting to find the optimum. Any integer value in the range 1 to 1000 is accepted. The default value is 50.
...	parameter catch all.

Details

The following parameters are supported:

Value

Returns a list of all validated parameters.

`navigate_hash` *navigate_hash generates DOT statements for all trees in current model.*

Description

This internal function generates DOT statements for the selected tree generated by the current model, grViz is then called to display the resultant graph.

Usage

```
navigate_hash(hobj, numtree = NA, dataset_description = "Unknown")
```

Arguments

hobj	List of all tree objects.
numtree	The number of the tree to display.
dataset_description	A brief description of the dataset being used.

Value

nothing.

NNode *NNode returns a tree node.*

Description

This internal function is used to create a tree node. Remaining parameters will be initialised as the tree induction process progresses.

Usage

```
NNode(
  gini,
  num_samples,
  num_samples_per_class,
  predicted_class,
  parent_node,
  objectid,
  oob_row_indices = NA
)
```

Arguments

gini	The gini index of the node.
num_samples	The number of samples at this node.
num_samples_per_class	A table showing class distribution at this node.
predicted_class	If a leaf node, the predicted class of this node.
parent_node	The parent node object for this child node.
objectid	A unique id for this node object.
oob_row_indices	For a root node the out-of-bag indices from original training set used to create this tree.

Value

a tree node of type hash.

pendigits *Pen-Based Recognition of Handwritten Digits Dataset.*

Description

We create a digit database by collecting 250 samples from 44 writers. The samples written by 30 writers are used for training, cross-validation and writer dependent testing, and the digits written by the other 14 are used for writer independent testing. This database is also available in the UNIPEN format.

Usage

```
data(pendigits)
```

Format

A data frame with 10,992 rows and 16 variables:

cols1-16 All input attributes are integers in the range 0..100.

class The last column is the class which has been coded as follows : The last attribute is the class code 0..9

Details

We use a WACOM PL-100V pressure sensitive tablet with an integrated LCD display and a cordless stylus. The input and display areas are located in the same place. Attached to the serial port of an Intel 486 based PC, it allows us to collect handwriting samples. The tablet sends x and y tablet coordinates and pressure level values of the pen at fixed time intervals (sampling rate) of 100 miliseconds. These writers are asked to write 250 digits in random order inside boxes of 500 by 500 tablet pixel resolution. Subject are monitored only during the first entry screens. Each screen contains five boxes with the digits to be written displayed above. Subjects are told to write only inside these boxes. If they make a mistake or are unhappy with their writing, they are instructed to clear the content of a box by using an on-screen button. The first ten digits are ignored because most writers are not familiar with this type of input devices, but subjects are not aware of this. In our study, we use only (x, y) coordinate information. The stylus pressure level values are ignored. First we apply normalization to make our representation invariant to translations and scale distortions. The raw data that we capture from the tablet consist of integer values between 0 and 500 (tablet input box resolution). The new coordinates are such that the coordinate which has the maximum range varies between 0 and 100. Usually x stays in this range, since most characters are taller than they are wide. In order to train and test our classifiers, we need to represent digits as constant length feature vectors. A commonly used technique leading to good results is resampling the (x_t, y_t) points. Temporal resampling (points regularly spaced in time) or spatial resampling (points regularly spaced in arc length) can be used here. Raw point data are already regularly spaced in time but the distance between them is variable. Previous research showed that spatial resampling to obtain a constant number of regularly spaced points on the trajectory yields much better performance, because it provides a better alignment between points. Our resampling algorithm uses simple linear interpolation between pairs of points. The resampled digits are represented as a sequence of T

points (x_t, y_t)_t = 1^T, regularly spaced in arc length, as opposed to the input sequence, which is regularly spaced in time. So, the input vector size is 2*T, two times the number of points resampled. We considered spatial resampling to T=8,12,16 points in our experiments and found that T=8 gave the best trade-off between accuracy and complexity.

Source

https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

---

`perform_ccp_driver` *perform_ccp_driver.*

Description

This function performs CCP pruning on the supplied tree.

Usage

```
perform_ccp_driver(treeobjs)
```

Arguments

`treeobjs`	The current HHCARTR tree that has just been induced contained within a list() object. CCP pruning will be performed on this tree.

Value

A dataframe containing information for each subtree.

`pima` *PIMA Indians diabetes dataset.*

Description

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Usage

```
data(pima)
```

Format

A data frame with 768 rows and 8 variables:

Pregnancies Number of times pregnant

Glucose Plasma glucose concentration a 2 hours in an oral glucose tolerance test

BloodPressure Diastolic blood pressure (mm Hg)

SkinThickness Triceps skin fold thickness (mm)

Insulin 2-Hour serum insulin (mu U/ml)

BMI Body mass index (weight in kg/(height in m)^2)

DiabetesPedigreeFunction Diabetes pedigree function

Age Age (years)

Outcome Outcome: target variable, whether patient had diabetes, 268 of 768 are 1, the others are 0 (1 = yes; 0 = no)

Details

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Source

https://www.kaggle.com/uciml/pima-indians-diabetes-database

`predict.hhcartr` *predict - Create generic S3method to make predictions via predict.hhcartr. Needs export entry in the NAMESPACE file.*

Description

This function creates a generic S3method predict which is used to call predict.hhcartr when an object of type hhcartr passed to the predict function, i.e. an object that is returned from the fit() function. The object created from the predict function supports the accuracy and predictions methods. The accuracy method returns the accuracy achieved on the test_data and the method predictions returns the actual predictions made on the test_data.

Usage

```
## S3 method for class 'hhcartr'
predict(object, ..., test_data)
```

Arguments

`object`	Unused parameter.
`...`	Unused parameter.
`test_data`	The test dataset the user wants to make predictions on.

Value

exposes the accuracy() and predictions() methods.

Examples

```
# source: /man/examples/predict.R

# Basic usage of predict().

# Note: we need to have a model to modify first.

# load our data.
X <- iris[,1:4]
y <- iris[,5]

# instantiate our model.
clf = HHDecisionTree(n_folds=1,
                     n_trees=1,
                     pruning=FALSE,
                     min_node_impurity=0.0)

# describe what dataset our model is using.
setDataDescription("IRIS Dataset")

# train our model.
model_output <- clf$fit(X, y)

# make predictions on an unseen test set.
# As the IRIS dataset provides no separate test set,
# for the sake of this example we will predict on the original data
# and pretend it is previously unseen data.
preds <- predict(model_output, test_data = iris)

# The predict object 'preds' exposes the following methods
# that we can use to extract the results of interest
# (depending upon model options used):

# preds$accuracy and preds$predictions.
```

`print.hhcartr` *print.hhcartr - Create generic S3method to print results via print.hhcartr. Needs export entry in the NAMESPACE file.*

Description

This function will generate a ggplot showing test set accuracy for each tree fold/trial. In future will need to provide support to allow user to customize the plots.

Usage

```
## S3 method for class 'hhcartr'
print(x, ...)
```

Arguments

x	Unused parameter.
...	Unused parameter.

Value

A ggplot of training accuracy.

Examples

```
# source: /man/examples/print.R

# Basic usage of print().

# Note: we need to have a model first.

# load our data.
X <- iris[,1:4]
y <- iris[,5]

# instantiate our model.
clf = HHDecisionTree(n_folds=10,
                     n_trees=1,
                     pruning=FALSE,
                     min_node_impurity=0.0)

# describe what dataset our model is using.
setDataDescription("IRIS Dataset")

# train our model.
model_output <- clf$fit(X, y)

# print fold accuracy distribution from our training data.
print(model_output)
```

---

prune.control *prune.control*

Description

This internal function is used to validate the parameters specified on the prune_control parameter.

Usage

```
prune.control(
  prune_type = "ccp",
  prune_stochastic_max_nodes = 10,
  prune_stochastic_max_depth = 10,
  prune_stochastic_samples = 100,
  ...
)
```

Arguments

prune_type	The prune type required, valid values are 'all', 'ccp' and stochastic'. The default value is 'all'.
prune_stochastic_max_nodes	The prune_stochastic_max_nodes parameter specifies the number of internal nodes to randomly sample on each prune_stochastic_samples. The value specified must be an even number as an equal number of left and right internal nodes will form the sample. The prune_stochastic_max_nodes parameter can have a value of any even integer in the range two to twenty-four. The default value is 10.
prune_stochastic_max_depth	When sampling internal nodes, the prune_stochastic_max_depth parameter specifies the maximum decision tree depth to select internal nodes from. Internal nodes occurring in the original decision tree at depths greater than prune_stochastic_max_depth are not eligible for sampling. Any positive integer in the range two to the maximum depth of the current tree is accepted. The default value is 12.
prune_stochastic_samples	The prune_stochastic_samples parameter specifies the number of times internal nodes will be sampled from the current decision tree. The number of internal nodes to be sampled each iteration is determined by the prune_stochastic_max_nodes parameter, the internal nodes eligible to be sampled is determined by the prune_stochastic_max_depth parameter. The prune_stochastic_samples parameter can have any positive integer value greater than zero. The default value is 100.
...	parameter catch-all.

Details

The following parameters are supported:

Value

Returns a list of validated parameters.

`pruning_make_predictions` *pruning_make_predictions Make predictions for each test dataset row against each tree.*

Description

This internal function is a front-end to function make_predictions for making predictions on CCP generated subtrees.

Usage

```
pruning_make_predictions(
  loop_count,
  j,
  alpha_df,
  current_tree,
  test,
  useIdentity,
  classify,
  colname = "collapse_this_node",
  pred_type = "fatbears"
)
```

Arguments

`loop_count`	The current fold number.
`j`	The current tree number.
`alpha_df`	A dataframe containing the CCP generated subtrees.
`current_tree`	The current decision tree in the hhcartr internal format.
`test`	The test dataset.
`useIdentity`	Whether the training data has been transformed with the householder transform.
`classify`	Default is TRUE. Set TRUE for a classification problem and FALSE for a regression problem.
`colname`	The column name in alpha_df that contains the list of internal node numbers that will be collapsed in the current subtree. The default value is "collapse_this_node".
`pred_type`	Is pruning_make_predictions being called with CCP generated subtrees or fatbears generated subtrees. Can have a value of either "ccp" or "fatbears". The default value is "fatbears".

Value

Returns a list of (new_df, subtree_accuracy_predictions_df).

reflect_feature_space *reflect_feature_space*

Description

This function is in an internal only function. It applies the householder transformation to the feature data.

Usage

```
reflect_feature_space(
  X_matrix,
  X,
  y,
  most_freq_class,
  n_classes,
  max_features,
  n_features,
  depth,
  colx
)
```

Arguments

X_matrix	The current node to be tested for a split.
X	The feature variables of the current node being processed.
y	The target variable for the corresponding feature variables.
most_freq_class	The most frequently occurring class at the current node.
n_classes	The number of classes in the target variable.
max_features	The maximum number of features to consider in the current split.
n_features	The number of feature variables.
depth	The depth of the current tree.
colx	The eigenvector column to use.

Details

The following parameters are supported:

Value

Returns idxA, thrA, gidxA, X_houseA, newH_A if split is on reflected data, otherwise a list containing idx, thr, X, NULL, FALSE is returned when the split is made using the original data when the first eigen vector equals the first column of the identity matrix.

`results` *results - Create generic S3method to display results via results.hhcartr. Needs export entry in the NAMESPACE file.*

Description

This function creates a generic S3method results which is used to call results.hhcartr when an object of type hhcartr passed to the results function, i.e. an object that is returned from the fit() function. Parameters and return are the same for the results.hhcartr function.

Usage

```
results(x, ...)
```

Arguments

`x`	Unused parameter.
`...`	Unused parameter.

Value

Prints relevant information about accuracy from model training. Object returned from call to results exposes the accuracy() method, which can then be used to return accuracy information for each fold/trial of the training process. Also exposed is the margin() method, this returns the margin for each tree in each fold.

Examples

```
# source: /man/examples/results.R

# Basic usage of results().

# Note: we need to have a model to modify first.

# load our data.
X <- iris[,1:4]
y <- iris[,5]

# instantiate our model.
clf = HHDecisionTree(n_folds=1,
                     n_trees=1,
                     pruning=FALSE,
                     min_node_impurity=0.0)

# describe what dataset our model is using.
setDataDescription("IRIS Dataset")

# train our model.
model_output <- clf$fit(X, y)
```

```
# create our results() object.
res <- results(model_output)

# The results object 'res' exposes the following methods
# that we can use to extract the results of interest
# (depending upon model options used):

# res$accuracy, res$margin, res$mni_data,
# res$ccp_subtree_data, res$ccp_phase_data, res$ccp_predictions.
```

`row_predict` *row_predict Make predictions for a test dataset row against a tree.*

Description

This internal function is used to run a test dataset row through a tree to make a prediction.

Usage

```
row_predict(xnode, test_row, useIdentity, objectid)
```

Arguments

`xnode`	Root node of the current tree we are making predictions against.
`test_row`	Current row from the test dataset.
`useIdentity`	Whether the training data has been transformed with the householder transform.
`objectid`	A list of node numbers that will be 'pruned' ie. when making predictions if the tree node matches a node in objectid the tree node will be used to make the prediction rather than traversing any underlying nodes.

Value

A prediction for the test dataset row.

segment *Image Segmentation Dataset.*

Description

The instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel. Each instance is a 3x3 region.

Usage

```
data(segment)
```

Format

A data frame with 2310 rows and 19 variables:

region-centroid-col the column of the center pixel of the region.

region-centroid-row the row of the center pixel of the region.

region-pixel-count the number of pixels in a region = 9.

short-line-density-5 the results of a line extractoin algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.

short-line-density-2 same as short-line-density-5 but counts lines of high contrast, greater than 5.

vedge-mean measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.

vegde-sd measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.

hedge-mean measures the contrast of vertically adjacent pixels. Used for horizontal line detection.

hedge-sd measures the contrast of vertically adjacent pixels. Used for horizontal line detection.

intensity-mean the average over the region of (R + G + B)/3.

rawred-mean the average over the region of the R value.

rawblue-mean the average over the region of the B value.

rawgreen-mean the average over the region of the G value.

exred-mean measure the excess red: (2R - (G + B)).

exblue-mean measure the excess blue: (2B - (G + R)).

exgreen-mean measure the excess green: (2G - (R + B)).

value-mean 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics).

saturatoin-mean 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics).

hue-mean 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics).

Source

http://archive.ics.uci.edu/ml/datasets/image+segmentation

setDataDescription *setDataDescription Save brief description of the dataset used by current model.*

Description

This function setDataDescription updates global variable pkg.env$model_data_description with a brief description of the dataset being used to train the current model. It is later used in command output displays and plots.

Usage

```
setDataDescription(new_description)
```

Arguments

new_description	the new dataset description to be saved for the current model.

Value

nothing.

Examples

```
# source: /man/examples/setDataDescription.R

# Basic usage of setDataDescription().

# Note: we need to have a model to modify first.

# load our data.
X <- iris[,1:4]
y <- iris[,5]

# instantiate our model.
clf = HHDecisionTree(n_folds=1,
                     n_trees=1,
                     pruning=FALSE,
                     min_node_impurity=0.0)

# describe what dataset our model is using.
setDataDescription("IRIS Dataset")

# train our model.
vv <- clf$fit(X, y)
```

split_using_original_data *split_using_original_data*

Description

This function is in an internal only function. It performs an axis-parallel split on the original data.

Usage

```
split_using_original_data(
  X,
  y,
  most_freq_class,
  split_original,
  n_classes,
  max_features,
  depth
)
```

Arguments

X	The feature variables of the current node being processed.
y	The target variable for the corresponding feature variables.
most_freq_class	The most frequently occurring class at the current node.
split_original	A flag, TRUE if splitting on original data otherwise FALSE.
n_classes	The number of classes in the target variable.
max_features	The maximum number of features to consider in the current split.
depth	The depth of the current tree.

Details

The following parameters are supported:

Value

returns a list containing the following: idx, thr, X, NULL, FALSE.

survival *Haberman's Survival Dataset.*

Description

The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

Usage

```
data(survival)
```

Format

A data frame with 306 rows and 3 variables:

Age Age of patient at time of operation (numerical)

Year Patient's year of operation (year - 1900, numerical)

Positive Nodes Number of positive axillary nodes detected (numerical)

Survival status (class attribute) – 1 = the patient survived 5 years or longer – 2 = the patient died within 5 year

Source

https://archive.ics.uci.edu/ml/datasets/Haberman's+Survival

testbed *testbed - Test internal hhcartr functions.*

Description

This function allows the testing of internal hhcartr functions in a standalone fashion. Function testbed requires an export so that it can be used by the devtools::check() and devtools::test() functions. The currently supported values for parameter test_func are ["best_split_", "split_using_original_data", "reflect_feature_space", "hhcart_reflect_feature_space_g", "hhcartr_regressor_find_better_split"].

Usage

```
testbed(
  X,
  y,
  most_freq_class,
  split_original,
  n_classes,
```

```
  max_features,
  test_func,
  n_features,
  X_matrix
)
```

Arguments

X	Training data, the feature variables.
y	Training data, the target variable.
most_freq_class	The most frequent class in the target variable.
split_original	boolean to indicate whether to split on original data or reflected data.
n_classes	The number of classes in y.
max_features	The maximum number of features to use when training random forests.
test_func	The hhcartr function to be tested.
n_features	The number of feature columns in the training dataset.
X_matrix	The A-matrix - rows containing the most frequent class in the training dataset.

Value

Returns the output of the test_func to be tested (if it returns output).

---

vehicle *Statlog (Vehicle Silhouettes) Dataset.*

Description

The purpose is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles.

Usage

```
data(vehicle)
```

Format

A data frame with 846 rows and 19 variables:

COMPACTNESS (average perim)**2/area.

CIRCULARITY (average radius)**2/area.

DISTANCE CIRCULARITY area/(av.distance from border)**2.

RADIUS RATIO (max.rad-min.rad)/av.radius.

PR.AXIS ASPECT RATIO (minor axis)/(major axis).

MAX.LENGTH ASPECT RATIO (length perp. max length)/(max length).

SCATTER RATIO (inertia about minor axis)/(inertia about major axis).

ELONGATEDNESS area/(shrink width)**2.

PR.AXIS RECTANGULARITY area/(pr.axis length*pr.axis width).

MAX.LENGTH RECTANGULARITY area/(max.length*length perp. to this).

SCALED VARIANCE ALONG MAJOR AXIS (2nd order moment about minor axis)/area.

SCALED VARIANCE ALONG MINOR AXIS (2nd order moment about major axis)/area.

SCALED RADIUS OF GYRATION (mavar+mivar)/area.

SKEWNESS ABOUT MAJOR AXIS (3rd order moment about major axis)/sigma_min**3.

SKEWNESS ABOUT MINOR AXIS (3rd order moment about minor axis)/sigma_maj**3.

KURTOSIS ABOUT MINOR AXIS (4th order moment about major axis)/sigma_min**4.

KURTOSIS ABOUT MAJOR AXIS (4th order moment about minor axis)/sigma_maj**4.

HOLLOWS RATIO (area of hollows)/(area of bounding polygon).

class 4 classes, OPEL, SAAB, BUS, VAN.

Source

http://archive.ics.uci.edu/ml/datasets/statlog+(vehicle+silhouettes)

`wine` *Wine recognition data.*

Description

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. All 13 attributes are continuous. Distribution of classes: class 1 59x, class 2 71x, class 3 48x.

Usage

```
wine
```

Format

A data frame with 178 rows and 13 variables:

Alcohol

Malic acid

Ash

Alcalinity of ash

Magnesium

Total phenols

Flavanoids

Nonflavanoid phenols

Proanthocyanins

Color intensity

Hue

OD280/OD315 of diluted wines

Proline

Source

https://archive.ics.uci.edu/ml/datasets/Wine

Index