Package 'daltoolboxdp'

April 24, 2025

Title Python-Based Extensions for Data Analytics Workflows

Version 1.0.787

Description Provides Python-based extensions to enhance data analytics workflows, particularly for tasks involving data preprocessing and predictive modeling. Includes tools for data sampling, transformation, feature selection, balancing strategies (e.g., SMOTE), and model construction. These capabilities leverage Python libraries via the reticulate interface, enabling seamless integration with a broader machine learning ecosystem. Supports instance selection and hybrid workflows that combine R and Python functionalities for flexible and reproducible analytical pipelines. The architecture is inspired by the Experiment Lines approach, which promotes modularity, extensibility, and interoperability across tools. More information on Experiment Lines is available in Ogasawara et al. (2009) <doi:10.1007/978-3-642-02279-1_20>.

License MIT + file LICENSE

URL https://cefet-rj-dal.github.io/daltoolboxdp/, https://github.com/cefet-rj-dal/daltoolboxdp

BugReports https://github.com/cefet-rj-dal/daltoolboxdp/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports daltoolbox, leaps, FSelector, doBy, glmnet, smotefamily, reticulate, stats

Config/reticulate list(packages = list(list(package = ``scipy"), list(package = ``torch"), list(package = ``pandas"), list(package = ``numpy"), list(package = ``matplotlib"), list(package = ``scikit-learn")))

NeedsCompilation no

Author Eduardo Ogasawara [aut, ths, cre] (<https://orcid.org/0000-0002-0466-0626>), Diego Salles [aut, ths], Federal Center for Technological Education of Rio de Janeiro (CEFET/RJ) [cph] (CEFET/RJ)

Maintainer Eduardo Ogasawara <eogasawara@ieee.org>

Repository CRAN

Date/Publication 2025-04-24 11:40:23 UTC

Contents

`bal_oversampling` *Oversampling*

Description

Oversampling balances the class distribution of a dataset by increasing the representation of the minority class in the dataset. It wraps the smotefamily library.

Usage

```
bal_oversampling(attribute)
```

Arguments

`attribute`	The class attribute to target balancing using oversampling.

Value

A `bal_oversampling` object.

Examples

```
data(iris)
mod_iris <- iris[c(1:50,51:71,101:111),]

bal <- bal_oversampling('Species')
bal <- daltoolbox::fit(bal, mod_iris)
adjust_iris <- daltoolbox::transform(bal, mod_iris)
table(adjust_iris$Species)
```

bal_subsampling *Subsampling*

Description

Subsampling balances the class distribution of a dataset by reducing the representation of the majority class in the dataset.

Usage

```
bal_subsampling(attribute)
```

Arguments

attribute	The class attribute to target balancing using subsampling

Value

A bal_subsampling object.

Examples

```
data(iris)
mod_iris <- iris[c(1:50,51:71,101:111),]

bal <- bal_subsampling('Species')
bal <- daltoolbox::fit(bal, mod_iris)
adjust_iris <- daltoolbox::transform(bal, mod_iris)
table(adjust_iris$Species)
```

cla_gb *Gradient Boosting Classifier*

Description

Implements a classifier using the Gradient Boosting algorithm. This function wraps the GradientBoostingClassifier from Python's scikit-learn library.

Usage

```
cla_gb(
  attribute,
  slevels,
  loss = "log_loss",
  learning_rate = 0.1,
  n_estimators = 100,
```

```
  subsample = 1,
  criterion = "friedman_mse",
  min_samples_split = 2,
  min_samples_leaf = 1,
  min_weight_fraction_leaf = 0,
  max_depth = 3,
  min_impurity_decrease = 0,
  init = NULL,
  random_state = NULL,
  max_features = NULL,
  verbose = 0,
  max_leaf_nodes = NULL,
  warm_start = FALSE,
  validation_fraction = 0.1,
  n_iter_no_change = NULL,
  tol = 1e-04,
  ccp_alpha = 0
)
```

Arguments

attribute	Target attribute name for model building
slevels	Possible values for the target classification
loss	Loss function to be optimized ('log_loss', 'exponential')
learning_rate	Learning rate that shrinks the contribution of each tree
n_estimators	Number of boosting stages to perform
subsample	Fraction of samples to be used for fitting the individual base learners
criterion	Function to measure the quality of a split
min_samples_split	Minimum number of samples required to split an internal node
min_samples_leaf	Minimum number of samples required to be at a leaf node
min_weight_fraction_leaf	Minimum weighted fraction of the sum total of weights
max_depth	Maximum depth of the individual regression estimators
min_impurity_decrease	Minimum impurity decrease required for split
init	Estimator object to initialize the model
random_state	Random number generator seed
max_features	Number of features to consider for best split
verbose	Controls verbosity of the output
max_leaf_nodes	Maximum number of leaf nodes
warm_start	Whether to reuse solution of previous call

`validation_fraction`
: Proportion of training data to set aside for validation

`n_iter_no_change`
: Used to decide if early stopping will be used

`tol`
: Tolerance for early stopping

`ccp_alpha`
: Complexity parameter for cost-complexity pruning

Details

Tree Boosting

Value

A Gradient Boosting classifier object

`cla_gb` object

Examples

```
library(daltoolboxdp)
```

`fs` *Feature Selection*

Description

Feature selection is a process of selecting a subset of relevant features from a larger set of features in a dataset for use in model training. The FeatureSelection class in R provides a framework for performing feature selection.

Usage

```
fs(attribute)
```

Arguments

`attribute`
: The target variable.

Value

An instance of the FeatureSelection class.

Examples

```
#See ?fs_fss for an example of feature selection
```

fs_fss *Forward Stepwise Selection*

Description

Forward stepwise selection is a technique for feature selection in which attributes are added to a model one at a time based on their ability to improve the model's performance. It stops adding once the candidate addition does not significantly improve model adjustment. It wraps the leaps library.

Usage

```
fs_fss(attribute)
```

Arguments

attribute	The target variable.

Value

A fs_fss object.

Examples

```
data(iris)
myfeature <- daltoolbox::fit(fs_fss("Species"), iris)
data <- daltoolbox::transform(myfeature, iris)
head(data)
```

fs_ig *Information Gain*

Description

Information Gain is a feature selection technique based on information theory. It measures the information obtained for the target variable by knowing the presence or absence of a feature. It wraps the FSelector library.

Usage

```
fs_ig(attribute)
```

Arguments

attribute	The target variable.

Value

A fs_ig object.

Examples

```
data(iris)
myfeature <- daltoolbox::fit(fs_ig("Species"), iris)
data <- daltoolbox::transform(myfeature, iris)
head(data)
```

---

fs_lasso *Feature Selection using Lasso*

Description

Feature selection using Lasso regression is a technique for selecting a subset of relevant features. It wraps the glmnet library.

Usage

```
fs_lasso(attribute)
```

Arguments

attribute	The target variable.

Value

A fs_lasso object.

Examples

```
data(iris)
myfeature <- daltoolbox::fit(fs_lasso("Species"), iris)
data <- daltoolbox::transform(myfeature, iris)
head(data)
```

fs_relief *Relief*

Description

Feature selection using Relief is a technique for selecting a subset of relevant features. It calculates the relevance of a feature by considering the difference in feature values between nearest neighbors of the same and different classes. It wraps the FSelector library.

Usage

```
fs_relief(attribute)
```

Arguments

attribute	The target variable.

Value

A fs_relief object.

Examples

```
data(iris)
myfeature <- daltoolbox::fit(fs_relief("Species"), iris)
data <- daltoolbox::transform(myfeature, iris)
head(data)
```

Index