Package 'tspredit'

November 9, 2023

Title Time Series Prediction Integrated Tuning

Version 1.0.737

Description Prediction is one of the most important activities while working with time series. There are many alternative ways to model the time series. Finding the right one is challenging to model them. Most data-driven models (either statistical or machine learning) demand tuning. Setting them right is mandatory for good predictions. It is even more complex since time series prediction also demands choosing a data pre-processing that complies with the chosen model. Many time series frameworks have features to build and tune models. The package differs as it provides a framework that seamlessly integrates tuning data pre-processing activities with the building of models. The package provides functions for defining and conducting time series prediction, including data pre(post)processing, decomposition, tuning, modeling, prediction, and accuracy assessment. More information is available at Izau et al. <doi:10.5753/sbbd.2022.224330>.

License MIT + file LICENSE

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

Encoding UTF-8

RoxygenNote 7.2.3

Imports dplyr, stats, glmnet, smotefamily, leaps, FSelector, forecast, mFilter, KFAS, doBy, daltoolbox

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 Rebecca Salles [aut],
Cristiane Gea [aut],
Vitoria Birindiba [aut],
Carla Pacheco [aut],
Eduardo Bezerra [aut],
Esther Pacitti [aut],
Fabio Porto [aut],

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

Maintainer Eduardo Ogasawara <eogasawara@ieee.org>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2023-11-09 16:40:02 UTC

R topics documented:

bal_oversampling . . . 3
bal_subsampling . . . 3
cla_fs . . . 4
cla_fs_fss . . . 5
cla_fs_ig . . . 5
cla_fs_lasso . . . 6
cla_fs_relief . . . 7
fertilizers . . . 7
ts_aug_awareness . . . 8
ts_aug_awaresmooth . . . 9
ts_aug_flip . . . 9
ts_aug_jitter . . . 10
ts_aug_none . . . 11
ts_aug_shrink . . . 11
ts_aug_stretch . . . 12
ts_aug_wormhole . . . 13
ts_fil_ema . . . 14
ts_fil_hp . . . 14
ts_fil_kalman . . . 15
ts_fil_lowess . . . 16
ts_fil_ma . . . 17
ts_fil_none . . . 18
ts_fil_recursive . . . 18
ts_fil_seas_adj . . . 19
ts_fil_smooth . . . 20
ts_fil_spline . . . 21
ts_maintune . . . 21
ts_norm_none . . . 22

Index **24**

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_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

```
cla_fs(attribute)
```

Arguments

attribute	The target variable.

Value

An instance of the FeatureSelection class.

Examples

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

cla_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

```
cla_fs_fss(attribute)
```

Arguments

attribute	The target variable.

Value

A cla_fs_fss object.

Examples

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

cla_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

```
cla_fs_ig(attribute)
```

Arguments

attribute	The target variable.

Value

A cla_fs_ig object.

Examples

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

---

cla_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

```
cla_fs_lasso(attribute)
```

Arguments

attribute	The target variable.

Value

A cla_fs_lasso object.

Examples

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

cla_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

```
cla_fs_relief(attribute)
```

Arguments

attribute	The target variable.

Value

A `cla_fs_relief` object.

Examples

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

fertilizers *Fertilizers (Regression)*

Description

List of Brazilian fertilizers consumption of N, P2O5, K2O.

- brazil_n: nitrogen consumption from 1961 to 2020.
- brazil_p2o5: phosphate consumption from 1961 to 2020.
- brazil_k2o: potash consumption from 1961 to 2020.

Usage

```
data(fertilizers)
```

Format

list of fertilizers' time series.

Source

This dataset was obtained from the MASS library.

References

International Fertilizer Association (IFA): http://www.fertilizer.org.

Examples

```
data(fertilizers)
head(fertilizers$brazil_n)
```

---

ts_aug_awareness *Augmentation by awareness*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. Awareness reinforce recent data preferably.

Usage

```
ts_aug_awareness(factor = 1)
```

Arguments

factor	increase factor for data augmentation

Value

a ts_aug_awareness object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using awareness
augment <- ts_aug_awareness()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

ts_aug_awaresmooth *Augmentation by awareness smooth*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. Awareness Smooth reinforce recent data preferably. It also smooths noise data.

Usage

```
ts_aug_awaresmooth(factor = 1)
```

Arguments

factor	increase factor for data augmentation

Value

a ts_aug_awaresmooth object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using awareness
augment <- ts_aug_awaresmooth()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

ts_aug_flip *Augmentation by flip*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. Flip mirror the sliding observations relative to the mean of the sliding windows.

Usage

```
ts_aug_flip()
```

Value

a ts_aug_flip object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using flip
augment <- ts_aug_flip()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

---

ts_aug_jitter *Augmentation by jitter*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. jitter adds random noise to each data point in the time series.

Usage

```
ts_aug_jitter()
```

Value

a ts_aug_jitter object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using flip
```

```
augment <- ts_aug_jitter()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

---

ts_aug_none *no augmentation*

Description

Does not make data augmentation.

Usage

```
ts_aug_none()
```

Value

a ts_aug_none object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#no data augmentation
augment <- ts_aug_none()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

---

ts_aug_shrink *Augmentation by shrink*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. stretch does data augmentation by decreasing the volatility of the time series.

Usage

```
ts_aug_shrink(scale_factor = 0.8)
```

Arguments

scale_factor for shrink

Value

a ts_aug_shrink object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using flip
augment <- ts_aug_shrink()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

---

ts_aug_stretch *Augmentation by stretch*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. stretch does data augmentation by increasing the volatility of the time series.

Usage

```
ts_aug_stretch(scale_factor = 1.2)
```

Arguments

scale_factor for stretch

Value

a ts_aug_stretch object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using flip
augment <- ts_aug_stretch()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

---

ts_aug_wormhole *Augmentation by wormhole*

Description

Time series data augmentation is a technique used to increase the size and diversity of a time series dataset by creating new instances of the original data through transformations or modifications. The goal is to improve the performance of machine learning models trained on time series data by reducing overfitting and improving generalization. Wormhole does data augmentation by removing lagged terms and adding old terms.

Usage

```
ts_aug_wormhole()
```

Value

a ts_aug_wormhole object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#data augmentation using flip
augment <- ts_aug_wormhole()
augment <- fit(augment, xw)
xa <- transform(augment, xw)
ts_head(xa)
```

ts_fil_ema *Time Series Exponential Moving Average*

Description

Used to smooth out fluctuations, while giving more weight to recent observations. Particularly useful when the data has a trend or seasonality component.

Usage

```
ts_fil_ema(ema = 3)
```

Arguments

ema	exponential moving average size

Value

a ts_fil_ema object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]
# convert to sliding windows
ts <- ts_data(sin_data$y, 10)
ts_head(ts, 3)
summary(ts[,10])

# filter
filter <- ts_fil_ema(ema = 3)
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

ts_fil_hp *Hodrick-Prescott Filter*

Description

This filter eliminates the cyclical component of the series, performs smoothing on it, making it more sensitive to long-term fluctuations. Each observation is decomposed into a cyclical and a growth component.

Usage

```
ts_fil_hp(lambda = 100, preserve = 0.9)
```

Arguments

lambda	It is the smoothing parameter of the Hodrick-Prescott filter. Lambda = 100*(frequency)^2 Correspondence between frequency and lambda values annual => frequency = 1 // lambda = 100 quarterly => frequency = 4 // lambda = 1600 monthly => frequency = 12 // lambda = 14400 weekly => frequency = 52 // lambda = 270400 daily (7 days a week) => frequency = 365 // lambda = 13322500 daily (5 days a week) => frequency = 252 // lambda = 6812100
preserve	value between 0 and 1. Balance the composition of observations and applied filter. Values close to 1 preserve original values. Values close to 0 adopts HP filter values.

Value

a ts_fil_hp object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

# filter
filter <- ts_fil_hp(lambda = 100*(26)^2)  #frequency assumed to be 26
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

---

ts_fil_kalman *Kalman Filter*

Description

The Kalman filter is an estimation algorithm that produces estimates of certain variables based on imprecise measurements to provide a prediction of the future state of the system.

Usage

```
ts_fil_kalman(H = 0.1, Q = 1)
```

Arguments

H	variance or covariance matrix of the measurement noise. This noise pertains to the relationship between the true system state and actual observations. Measurement noise is added to the measurement equation to account for uncertainties or errors associated with real observations. The higher this value, the higher the level of uncertainty in the observations.
Q	variance or covariance matrix of the process noise. This noise follows a zero-mean Gaussian distribution. It is added to the equation to account for uncertainties or unmodeled disturbances in the state evolution. The higher this value, the greater the uncertainty in the state transition process.

Value

a `ts_fil_kalman` object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

# filter
filter <- ts_fil_kalman()
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

`ts_fil_lowess` *Lowess Smoothing*

Description

It is a smoothing method that preserves the primary trend of the original observations and is used to remove noise and spikes in a way that allows data reconstruction and smoothing.

Usage

```
ts_fil_lowess(f = 0.2)
```

Arguments

f	smoothing parameter. The larger this value, the smoother the series will be. This provides the proportion of points on the plot that influence the smoothing.

Value

a ts_fil_lowess object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

# filter
filter <- ts_fil_lowess(f = 0.2)
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

---

ts_fil_ma *Time Series Moving Average*

Description

Used to smooth out fluctuations and reduce noise in a time series.

Usage

```
ts_fil_ma(ma = 3)
```

Arguments

ma	moving average size

Value

a ts_fil_ma object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]
# convert to sliding windows
ts <- ts_data(sin_data$y, 10)
ts_head(ts, 3)
summary(ts[,10])

# filter
```

```
filter <- ts_fil_ma(3)
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

`ts_fil_none` *no filter*

Description

Does not make data filter

Usage

```
ts_fil_none()
```

Value

a `ts_fil_none` object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#no data filter
filtering <- ts_fil_none()
filtering <- fit(filtering, xw)
xa <- transform(filtering, xw)
ts_head(xa)
```

`ts_fil_recursive` *Recursive Filter*

Description

Applies linear filtering to a univariate time series or to each series within a multivariate time series. It is useful for outlier detection, and the calculation is done recursively. This recursive calculation has the effect of reducing autocorrelation among observations, so that for each detected outlier, the filter is recalculated until there are no more outliers in the residuals.

Usage

```
ts_fil_recursive(filter)
```

Arguments

filter	smoothing parameter. The larger the value, the greater the smoothing. The smaller the value, the less smoothing, and the resulting series shape is more similar to the original series.

Value

a ts_fil_recursive object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

# filter
filter <- ts_fil_recursive(filter =  0.05)
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

---

ts_fil_seas_adj *Seasonal Adjustment*

Description

Removes the seasonal component from the time series without affecting the other components.

Usage

```
ts_fil_seas_adj(frequency = NULL)
```

Arguments

frequency	Frequency of the time series. It is an optional parameter. It can be configured when the frequency of the time series is known.

Value

a ts_fil_seas_adj object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

# filter
filter <- ts_fil_seas_adj(frequency = 26)
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

`ts_fil_smooth` *Time Series Smooth*

Description

Used to remove or reduce randomness (noise).

Usage

```
ts_fil_smooth()
```

Value

a `ts_fil_smooth` object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

filter <- ts_fil_smooth()
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

ts_fil_spline *Smoothing Splines*

Description

Fits a cubic smoothing spline to a time series.

Usage

```
ts_fil_spline(spar = NULL)
```

Arguments

spar	smoothing parameter. When spar is specified, the coefficient of the integral of the squared second derivative in the fitting criterion (penalized log-likelihood) is a monotone function of spar. #'@return a ts_fil_spline object.

Examples

```
# time series with noise
library(daltoolbox)
data(sin_data)
sin_data$y[9] <- 2*sin_data$y[9]

# filter
filter <- ts_fil_spline(spar = 0.5)
filter <- fit(filter, sin_data$y)
y <- transform(filter, sin_data$y)

# plot
plot_ts_pred(y=sin_data$y, yadj=y)
```

ts_maintune *Time Series Tune*

Description

Time Series Tune

Usage

```
ts_maintune(
  input_size,
  base_model,
  folds = 10,
  preprocess = list(daltoolbox::ts_norm_gminmax()),
  augment = list(ts_aug_none())
)
```

Arguments

input_size	input size for machine learning model
base_model	base model for tuning
folds	number of folds for cross-validation
preprocess	list of preprocessing methods
augment	data augmentation method

Value

a ts_maintune object.

Examples

```
library(daltoolbox)
data(sin_data)
ts <- ts_data(sin_data$y, 10)

samp <- ts_sample(ts, test_size = 5)
io_train <- ts_projection(samp$train)
io_test <- ts_projection(samp$test)

tune <- ts_maintune(input_size=c(3:5), base_model = ts_elm(), preprocess = list(ts_norm_gminmax()))
ranges <- list(nhid = 1:5, actfun=c('purelin'))

# Generic model tunning
model <- fit(tune, x=io_train$input, y=io_train$output, ranges)

prediction <- predict(model, x=io_test$input[1,], steps_ahead=5)
prediction <- as.vector(prediction)
output <- as.vector(io_test$output)

ev_test <- evaluate(model, output, prediction)
ev_test
```

---

ts_norm_none *no normalization*

Description

Does not make data normalization.

Usage

```
ts_norm_none()
```

Value

a ts_norm_none object.

Examples

```
library(daltoolbox)
data(sin_data)

#convert to sliding windows
xw <- ts_data(sin_data$y, 10)

#no data normalization
normalize <- ts_norm_none()
normalize <- fit(normalize, xw)
xa <- transform(normalize, xw)
ts_head(xa)
```

Index

∗ **datasets**
fertilizers, 7

bal_oversampling, 3
bal_subsampling, 3

cla_fs, 4
cla_fs_fss, 5
cla_fs_ig, 5
cla_fs_lasso, 6
cla_fs_relief, 7

fertilizers, 7

ts_aug_awareness, 8
ts_aug_awaresmooth, 9
ts_aug_flip, 9
ts_aug_jitter, 10
ts_aug_none, 11
ts_aug_shrink, 11
ts_aug_stretch, 12
ts_aug_wormhole, 13
ts_fil_ema, 14
ts_fil_hp, 14
ts_fil_kalman, 15
ts_fil_lowess, 16
ts_fil_ma, 17
ts_fil_none, 18
ts_fil_recursive, 18
ts_fil_seas_adj, 19
ts_fil_smooth, 20
ts_fil_spline, 21
ts_maintune, 21
ts_norm_none, 22