Package 'STResamplingDSAA'

September 23, 2019

Version 0.5-1

Date 2015-01-01

Title Biased Resampling Strategies for Imbalanced Spatio-Temporal Forecasting

Description Code supporting the conference article: Oliveira M, Moniz N, Torgo L, Santos Costa V (2019). ``Biased Resampling Strategies for Imbalanced Spatio-Temporal Forecasting.'' In Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA).

Depends R (>= 3.1.0)

Imports assertthat (>= 0.2.0),
dplyr (>= 0.7.5),
DMwR2 (>= 0.0.2),
foreach (>= 1.4.4),
lubridate (>= 1.7.4),
lwgeom (>= 0.1-3),
sf (>= 0.6-0),
stringr (>= 1.3.1),
uba (>= 0.7.8),
UBL (>= 0.0.6)

Suggests doParallel,
earth (>= 4.6.3),
ranger (>= 0.9.0),
rpart (>= 4.1-13),
scmamp,
ggplot2

License GPL (>=2)

RoxygenNote 6.1.1

R topics documented:

`add_ratios` *Add ratios between spatio-temporal neighborhood indicators*

Description

Add ratios between spatio-temporal neighborhood indicators

Usage

```
add_ratios(df, var, indStat = "mean")
```

Arguments

df	A data frame of spatio-temporal indicators. Column names should be of type <variable name>_<indStat>_<radius>.
var	A character string with the name of the variable with indicators to add ratios
indStat	The name of the summarizing stat that was used to calculate the indicators

Value

A data frame including the original data of df and additional ratios between the indicators of subsequent radiuses.

See Also

get_st_indicators

---

centralImputNAs *Handling NAs in train and test*

Description

Discard columns/rows with too many NAs and then impute with central value.

Usage

```
centralImputNAs(train, test, nORp)
```

Arguments

train	training data set
test	testing data set
nORp	minimum percentage of NA values in a row/column for that row/column to be discarded from training set

Value

list with an entry for the training and test sets (in this order), both now with no NA values

cv_folds *Cut into folds*

Description

Assigns rows of a data frame into folds for cross-validation.

Usage

```
cv_folds(x, nfolds)
```

Arguments

x	a data.frame
nfolds	number of folds

Value

a vector with the fold assignment of each row

df2site_sf *Create an sf object of available sites*

Description

Extracts the location information from a data frame and transforms into a `sf` object.

Usage

```
df2site_sf(df, site_id, lon, lat, crs)
```

Arguments

df	a data frame of the data set
site_id	the name of the column containing location IDs
lon	the name of the column containing the location's longitude
lat	the name of the column containing the location's latitude
crs	the code for the Coordinate Reference System

Value

a sf object, containing the geographic information for each location in `df`

See Also

`st_as_sf`

embed_series *Embed each time series in a spatio-temporal data set*

Description

Embed each time series in a spatio-temporal data set

Usage

```
embed_series(df, var, k, time = "time", station_id = "station")
```

Arguments

df	data frame
var	a character string, the name of the variable to embed
k	a numeric, the embed size
time	a character string, the column name identifying the time of observation
station_id	a character string, the column name identifying the location of observation

Value

A data frame with extra columns var_Tm1, var_Tm2, ..., var_Tm\(k-1\)

estimates *Estimate error using a chosen method*

Description

Estimate error using a chosen method

Usage

```
estimates(data, form, estimator = "kf_xval", est.pars = list(nfolds =
  10, fold.alloc.proc = "Trand_SPrand"), workflow = "simple_workflow",
  wf.pars = NULL, evaluator = "evaluate", eval.pars = NULL,
  seed = 1234)
```

Arguments

data	a data frame
form	a formula for learning
estimator	the name of an error estimator function
est.pars	a named list of arguments to feed to estimator
workflow	the name of the workflow to use for making predictions
wf.pars	a named list of arguments to feed to workflow
evaluator	the name of the function to use to calculate evaluation results
eval.pars	a named list of arguments to feed to evaluator
seed	a seed to set before performing estimates

Value

The results of evaluator after applying estimator to the learning task

---

evaluate *Evalute the results of a predictive workflow*

Description

Calculate evaluation metrics from the raw results of a workflow

Usage

```
evaluate(wfRes, eval.function = get("regressionMetrics",
  asNamespace("performanceEstimation")), .keptTrain = TRUE, ...)
```

Arguments

wfRes	a data frame (or list of data frames) containing the results of a predictive workflow with columns trues and preds containing the real and predicted values, respectively
eval.function	the function to be used to calculate error metrics from wfRes
.keptTrain	a Boolean indicating whether .keepTrain was set to TRUE in calls to estimation methods. Only useful if evaluation metrics need training data.
...	parameters to pass to eval.function

Value

The results (or a list of results) of eval.function applied to the data frame (or list of data frames) in wfRes

eval_stats *Calculate regression evaluation metrics*

Description

Calculate regression metrics for imbalanced domains.

Usage

```
eval_stats(trues, preds, y_train, cf = 1.5, thr = 0.9, beta = 1)
```

Arguments

trues	a vector of true values
preds	a vector of predicted values
y_train	a vector of training values
cf	phi.control coef
thr	relevance threshold
beta	beta in F-measure

Value

a named vector of metrics RMSE, relevance-aware RMSE, and utility-based precision, recall and F-measure

Author(s)

Nuno Moniz

get_all_neib_vals *Get the spatio-temporal neighbourhoods of all observations*

Description

Get the spatio-temporal neighbourhoods of all observations

Usage

```
get_all_neib_vals(df, max_radius, t_dist_mat, s_dist_mat, alpha, vars,
  time_id, site_id, parallel = FALSE, nsplits = 4)
```

Arguments

df	a data frame of observations
max_radius	the maximum spatio-temporal distance allowed to be included in a neighbourhood
t_dist_mat	a matrix of normalized temporal distances between time-stamps (rownames and colnames should be a concatenation of "TIME_" and the time-stamp)
s_dist_mat	a matrix of normalized spatial distances between locations (rownames and colnames should be a concatenation of "SITE_" and the location IDs)
alpha	a weighting factor for the spatio-temporal distance
vars	Vector of character strings indicating the columns whose values should be retrieved
time_id	the name of the column containing time-stamps
site_id	the name of the column containing location IDs
parallel	Boolean indicating whether the code should run in parallel. Default is FALSE
nsplits	Number of subsets of rows to split the data frame into so they can be processed in parallel

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between timestapms and α is a weighting factor. Note that the `radius` should always be a number between zero and `min(alpha,alpha-1)`. Also note that if `alpha` is set to 1, then instead of a cone, the neighbourhood will have the shape of a cylinder.

Value

A data frame where each row describes a neighbour, with the first two columns containing the location ID and time-stamp of the central observation, followed by two columns with the neighbouring location ID and time-stamp, a colunm containing the spatio-temporal distance between the two, and a final column containing the values of the variables in df at the neighbouring time and location.

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.8742&rep=rep1&type=pdf

See Also

get_st_neighbours

`get_full_indicators` *Get time series embeds and spatio-temporal indicators*

Description

Get time series embeds and spatio-temporal indicators

Usage

```
get_full_indicators(df, stations, k, betas, alpha = 0.5, var = "value",
  stats = c("mean", "weighted.mean", "sd"), ratios2add = c(TRUE, TRUE,
  FALSE), neib_type = "cone", parallel = FALSE, nsplits = 1,
  time_id = "time", site_id = "station")
```

Arguments

`df`	A data frame containing spatio-temporal information
`stations`	An `sf` object containing geographical information on the location of `df`
`k`	A numeric indicating the temporal embed size \(number\)
`betas`	A vector of values defining the maximum spatio-temporal distance allowed for an observation to be considered within a spatio-temporal neighbourhood
`alpha`	a weighting factor for the spatio-temporal distance
`var`	The name of the variable to summarize into indicators
`stats`	A vector containing the names of functions that are to be used to calculate summarizing statistics
`ratios2add`	A vector of Boolean values indicating, for each statistic in `stats` whether ratios between neighorhoods of subsequent sizes should be included as extra columns
`neib_type`	the type of neighborhood to consider. Can be • `cone` (default) - a cone with the center of its base at the observation (spatial radius growing with time) • `reversed` - a cone with its peak at the observation (spatial radius shrinking with time)
`parallel`	Boolean indicating whether the code should run in parallel. Default is FALSE
`nsplits`	Number of subsets of rows to split the data frame into so they can be processed in parallel
`time_id`	The name of the column containing time-stamps in `df`
`site_id`	The name of the column containing location IDs in `df`

Value

A data frame that contains extra columns <var>_Tm1, <var>_Tm2, ..., <var>_Tm<k-1> with previous observations for each location, summary statistics of the values of `var` found within the spatio-temporal neighbourhoods of the one or more `radiuses` of each pair (location ID, time-stamp) and ratios between them

get_phi *Calculate utility-based relevance*

Description

Calculate relevance of values given a parametrization of the relevance function. Most relevant: phi -> 1; less relevant: phi -> 0.

Usage

```
get_phi(y, phi.control)
```

Arguments

y	vector of values to calculate relevance of
phi.control	list of parameters as returned by function UBL::phi.control

See Also

phi, phi.control

get_space_wts *Calculate spatially-biased re-sampling weights*

Description

Calculate weights for re-sampling with a spatial bias. Observations have a distance that tends to 1 as they are farther away from the closest relevant case (besides itself) at time slice t (meaning they are more likely to be kept). Farthest away from relevant cases at time slice t: d -> 1.

Usage

```
get_space_wts(df, phi, rel.thr, sites_sf = NULL, lon = NULL,
  lat = NULL, crs = NULL, site_id, time)
```

Arguments

df	a data frame
phi	a vector of the relevance values of df's target variable
rel.thr	a relevance threshold above which an observation is considered relevant
sites_sf	An sf obejct containing station and IDs and geometry points of the locations. As an alternative, provide lon, lat, and crs
lon	the name of the column containing the location's longitude
lat	the name of the column containing the location's latitude

crs	the code for the Coordinate Reference System
site_id	the name of the column containing location IDs
time	the column name of the time-stamp

Value

A vector of spatially-biased re-sampling weights, scaled to fit within range [0,1].

---

get_spatial_dist_mat *Calculate spatial distance matrix*

Description

A function that calculates the geographical distance matrix between the locations of an sf object.

Usage

```
get_spatial_dist_mat(sites_sf, site_id)
```

Arguments

sites_sf	an sf object with the geograhic information of the locations (as returned by df2site_sf)
site_id	the column name of the location ID

Value

a matrix of distances. Row and column names are a concatenation of "SITE_" and the location IDs.

See Also

df2site_sf

---

get_st_indicator *Get a spatio-temporal indicator from neighbourhood data frame*

Description

Calculate a spatio-temporal indicator of a certain variable within a spatio-temporal neighborhood of a certain radius.

Usage

```
get_st_indicator(all_neib_vals, stat, radius, ind_name, var,
  time_id = "time", site_id = "site")
```

Arguments

all_neib_vals	a data frame containing information on observations spatio-temporal distance to neighbours and variable values at the neighbouring locations and times.
stat	the name of a function that calculates a statistic (e.g., "mean"). If the stat is "weighted.mean" then the inverse of the spatio-temporal distance is used to weight the values of observations in the spatio-temporal neighbourhood
radius	a value defining the maximum spatio-temporal distance allowed for an observation to be considered within a spatio-temporal neighbourhood
ind_name	the name of the indicator column
var	the name of the variable to summarize into an indicator
time_id	the name of the column containing time-stamps
site_id	the name of the column containing location IDs

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j}x\alpha + t_{i,j}x(1-\alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between time-stapms and α is a weighting factor. Note that `radius` should always be a number between zero and `min(alpha,alpha-1)`, so the border conditions apply. Also note that if `alpha` is set to 1, then instead of a cone, the neighbourhood will have the shape of a cylinder.

Value

A data frame that contains a summary statistic of the values found within the spatio-temporal neighbourhood of a certain radius of each pair (location ID, time-stamp) in `all_neib_vals`

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.8742&rep=rep1&type=pdf

---

get_st_indicators *Get spatio-temporal indicators from a data frame containing spatio-temporal information*

Description

Calculate spatio-temporal indicators of one or more variables within a spatio-temporal neighborhood of one or more maximum radius (in terms of spatio-temporal distance).

Usage

```
get_st_indicators(df, stations_sf, radiuses = c(0.1), stats = c("mean",
  "sd"), alpha = 0.5, neib_type = "cone", time_id = "time",
  site_id = "site_id", vars = c("value"), parallel = FALSE,
  nsplits = 4)
```

Arguments

df	A data frame containing spatio-temporal information
stations_sf	An sf object containing geographical information on the location of df
radiuses	A vector of values defining the maximum spatio-temporal distance allowed for an observation to be considered within a spatio-temporal neighbourhood
stats	A vector containing the names of functions that are to be used to calculate summarizing statistics
alpha	a weighting factor for the spatio-temporal distance
neib_type	the type of neighborhood to consider. Can be • cone (default) - a cone with the center of its base at the observation (spatial radius growing with time) • reversed - a cone with its peak at the observation (spatial radius shrinking with time)
time_id	The name of the column containing time-stamps in df
site_id	The name of the column containing location IDs in df
vars	The name of the variables to summarize into indicators
parallel	Boolean indicating whether the code should run in parallel. Default is FALSE
nsplits	Number of subsets of rows to split the data frame into so they can be processed in parallel

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between time-stapms and α is a weighting factor.

Value

A data frame that contains summary statistics of the values of vars found within the spatio-temporal neighbourhoods of the one or more radiuses of each pair (location ID, time-stamp) in df

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.8742&rep=rep1&type=pdf

`get_st_neighbours` *Get spatio-temporal neighbourhood*

Description

A function that calculates the observations that are within a spatio-temporal neighbourhood of a certain radius of a time and location.

Usage

```
get_st_neighbours(site, time, radius, t_dist_mat, s_dist_mat, alpha,
  time_id = "time", site_id = "site_id")
```

Arguments

`site`	a location ID
`time`	a time-stamp
`radius`	a radius of spatio-temporal distance
`t_dist_mat`	a matrix of normalized temporal distances between time-stamps (rownames and colnames should be a concatenation of "TIME_" and the time-stamp)
`s_dist_mat`	a matrix of normalized spatial distances between locations (rownames and colnames should be a concatenation of "SITE_" and the location IDs)
`alpha`	a weighting factor for the spatio-temporal distance
`time_id`	the name to give to the column of time-stamps (Default: time)
`site_id`	the name to give to the column of location IDs (Default: site_id)

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between time-stapms and α is a weighting factor.

Note that `radius` should always be a number between zero and `min(alpha,alpha-1)`. Also note that if `alpha` is set to 1, then instead of a cone, the neighbourhood will have the shape of a cylinder.

Value

A data frame where each row describes a neighbour, with the first two columns containing the location ID and time-stamp of the central observation, followed by two columns with the neighbouring location ID and time-stamp, and a final colunm containing the spatio-temporal distance between the two.

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.8742&rep=rep1&type=pdf

get_time_dist_mat *Calculate temporal distance matrix*

Description

A function that calculates a distance matrix of time-stamps

Usage

```
get_time_dist_mat(times, origin = min(times))
```

Arguments

times	A vector of time-stamps
origin	A date to use as origin for difftime

Value

a matrix of distances. Row and column names are a concatenation of "TIME_" and the time-stamp.

get_time_wts *Calculate temporally-biased re-sampling weights*

Description

Calculate weights for re-sampling with a temporal bias. Most recent observations have weights that tend to 1, while the oldest observations have weights that tend to 0 (meaning they are less likely to be kept). Most recent observations: w -> 1; oldest: w -> 0.

Usage

```
get_time_wts(times, phi, rel.thr)
```

Arguments

times	a vector of time-stamps
phi	a vector of the relevance values of df's target variable
rel.thr	a relevance threshold above which an observation is considered relevant

Value

A vector of temporally-biased re-sampling weights, scaled to fit within range [0,1].

Author(s)

Mariana Oliveira

`internal_workflow` *A learning and prediction workflow with internal validation*

Description

A learning and prediction workflow that may deal with NAs and use internal validation to parametrize a re-sampling technique to balance an imbalanced regression problem.

Usage

```
internal_workflow(train, test, form, model = "lm", resample = NULL,
  resample.pars = NULL, internal.est = NULL,
  internal.est.pars = NULL, internal.evaluator = NULL,
  internal.eval.pars = NULL, metrics = NULL, metrics.max = NULL,
  stat = "MED", resample.grid = NULL, handleNAs = NULL,
  min_train = 2, nORp = 0.2, time = "time", site_id = "site",
  .full_intRes = FALSE, ...)
```

Arguments

`train`	a data frame for training
`test`	a data frame for testing
`form`	a formula describing the model to learn
`model`	the name of the algorithm to use
`resample`	re-sampling technique to be used. Default is NULL.
`resample.pars`	parameters to be passed to re-sample function. Default is NULL.
`internal.est`	character string identifying the internal estimator function to use
`internal.est.pars`	named list of internal estimator parameters (e.g., tr.perc or nfolds)
`internal.evaluator`	character string indicating internal evaluation function
`internal.eval.pars`	named list of parameters to feed to internal evaluation function
`metrics`	vector of names of two metrics to be used to determine the best parametrization (the second metric is only used in case of ties)
`metrics.max`	vector of Booleans indicating whether each metric in parameter metrics should be maximized (TRUE) or minimized (FALsE) for best results
`stat`	parameter indicating summary statistic that should be used to determine the best internal evaluation metric: "MED" (for median) or "MEAN" (for mean)
`resample.grid`	a data.frame with columns indicating resample.pars to test using internal.est
`handleNAs`	string indicating how to deal with NAs. If "centralImput", training observations with at least 80% of non-NA columns, will have their NAs substituted by the mean value and testing observatiosn will have their NAs filled in with mean value regardless. Default is NULL.

min_train	a minimum number of observations that must be left to train a model. If there are not enough observations, predictions will be NA. Default is 2.
nORp	a maximum number or fraction of columns/rows with missing values above which a row/column will be removed from train before learning the model. Only works if handleNAs was set to centralImputation. Default is 0.2.
time	the name of the column in train and test containing time-stamps
site_id	the name of the column in train and test containing location IDs
.full_intRes	a Boolean indicating whether the full results object for internal validation should be returned as well. Defaults to FALSE
...	other parameters to feed to model

Value

a data frame containing time-stamps, location IDs, true values and predicted values

---

int_util_evaluate *Evalute the results of an internal predictive workflow*

Description

Calculate evaluation metrics from the raw results of a workflow

Usage

```
int_util_evaluate(wfRes, eval.function = get("regressionMetrics",
  asNamespace("performanceEstimation")), y_train, ...)
```

Arguments

wfRes	a data frame (or list of data frames) containing the results of a predictive workflow with columns trues and preds containing the real and predicted values, respectively
eval.function	the function to be used to calculate error metrics from wfRes
y_train	a vector of the whole training values
...	parameters to pass to eval.function

Value

The results (or a list of results) of eval.function applied to the data frame (or list of data frames) in wfRes

kf_xval *Cross-validation*

Description

Performs a cross-validation experiment where folds can be allocated in different ways considering time and/or space

Usage

```
kf_xval(data, nfolds, FUN, form, fold.alloc.proc = "Trand_SPrand",
  alloc.pars = NULL, time = "time", site_id = "site",
  .keepTrain = TRUE, .parallel = TRUE, ...)
```

Arguments

data	full dataset
nfolds	number of folds for the data set to be separated into. If you would like to set the number of time and space folds separately, nfolds should be set to NULL and t.nfolds and sp.nfolds should be fed as a list to alloc.pars (only available when using fold.alloc.proc set to Tblock_SPrand).
FUN	function with arguments • train training set • test testing set • time column name of time-stamps • site_id column name of location identifiers • form a formula for model learning • ... other arguments
form	a formula for model learning
fold.alloc.proc	name of fold allocation function. Should be one of • Trand_SPrand – each fold contains completely random observations. The default • Tall_SPrand - each fold includes all time and random locations in space • Tblock_SPall - each fold includes a block of contiguous time for all locations • Trand_SPall - each fold includes random time-snapshots of of all locations • Tblock_SPrand - each fold includes a block of contiguous time for a randomly assigned part of space
alloc.pars	parameters to pass onto fold.alloc.proc
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site_id"

.keepTrain	if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.
.parallel	Boolean indicating whether each fold should be run in parallel
...	other arguments to FUN

Value

The results of FUN. Usually, a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

---

norm_scale *Feature scaling*

Description

Normalize values to be within the range between [0,1].

Usage

```
norm_scale(x)
```

Arguments

x	a vector of values

Value

a scaled vector

---

prequential_eval *Prequential evaluation*

Description

Performs an evaluation procedure where training and test sets can be allocated in different ways, while always respecting the ordering provided by time (models are trained in the past and tested in the relative future).

Usage

```
prequential_eval(data, nfolds, FUN, form, window = "growing",
  fold.alloc.proc = "Tblock_SPall", alloc.pars = NULL,
  removeSP = FALSE, init_fold = 2, time = "time", site_id = "site",
  .keepTrain = TRUE, .parallel = TRUE, ...)
```

Arguments

data	full dataset
nfolds	number of folds for the data set to be separated into. If you would like to set the number of time and space folds separately, nfolds should be set to NULL and t.nfolds and sp.nfolds should be fed as a list to alloc.pars (only available when using fold.alloc.proc Tblock_SPrand)
FUN	function with arguments • train training set • test testing set • time column name of time-stamps • site_id column name of location identifiers • form a formula for model learning • ... other arguments
form	a formula for model learning
window	type of blocked-time window ordering considered. Should be one of • growing - for each time block being tested, all previous time blocks are used for training • sliding - for each time block being tested, the immediately previous time blocks are used for training
fold.alloc.proc	name of fold allocation function. Should be one of • Tblock_SPall - each fold includes a block of contiguous time for all locations • Tblock_SPrand - each fold includes a block of contiguous time for a randomly assigned part of space
alloc.pars	parameters to pass onto fold.alloc.proc
removeSP	argument that determines whether spatio-temporal blocks including the space being used for testing should be removed from the training set. Default is FALSE, meaning the information is not removed
init_fold	first temporal fold to use for testing. Default is 2.
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site_id"
.keepTrain	if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.
.parallel	Boolean indicating whether each block should be run in parallel
...	other arguments to FUN

Value

The results of FUN. Usually, a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

RandOverRegress *Random over-sampling for imbalanced regression problems*

Description

This function is identical to the function of the same name available on the R package UBL. The only difference is in the requirements imposed on the argument C.perc.
This function performs a random over-sampling strategy for imbalanced regression problems. Basically a percentage of cases of the "class(es)" (bumps above a relevance threshold defined) selected by the user are randomly over-sampled. Alternatively, it can either balance all the existing "classes" (the default) or it can "smoothly invert" the frequency of the examples in each class.

Usage

```
RandOverRegress(form, dat, rel = "auto", thr.rel = 0.5,
  C.perc = "balance", repl = TRUE)
```

Arguments

form	A formula describing the prediction problem.
dat	A data frame containing the original imbalanced data set.
rel	The relevance function which can be automatically ("auto") determined (the default) or may be provided by the user through a matrix with the interpolating points.
thr.rel	A number indicating the relevance threshold above which a case is considered as belonging to the rare "class".
C.perc	A vector containing the over-sampling percentage/s to apply to all/each "class" (bump) obtained with the relevance threshold. Replicas of the examples are are randomly added in each "class". If only one percentage is provided this value is reused in all the "classes" that have values above the relevance threshold. A different percentage can be provided to each "class". In this case, the percentages should be provided in ascending order of target variable value. The over-sampling percentage(s), should be numbers above 0, meaning that the important cases (cases above the threshold) are over-sampled by the corresponding percentage. If the number 1 is provided then the number of extreme examples will be doubled. Alternatively, C.perc parameter may be set to "balance" or "extreme", cases where the over-sampling percentages are automatically estimated to either balance or invert the frequencies of the examples in the "classes" (bumps).
repl	A boolean value controlling the possibility of having repetition of examples when choosing the examples to repeat in the over-sampled data set. Defaults to TRUE because this is a necessary condition if the selected percentage is greater than 2. This parameter is only important when the over-sampling percentage is between 1 and 2. In this case, it controls if all the new examples selected from a given "class" can be repeated or not.

Details

The only difference between this function and the original function is in the requirements imposed on the argument C.perc.
This function performs a random over-sampling strategy for dealing with imbalanced regression problems. The new examples included in the new data set are randomly selected replicas of the examples already present in the original data set.

Value

The function returns a data frame with the new data set resulting from the application of the random over-sampling strategy.

References

Paula Branco, Rita P. Ribeiro, Luis Torgo (2016)., UBL: an R Package for Utility-Based Learning, CoRR abs/1604.08079 [cs.MS], URL: http://arxiv.org/abs/1604.08079

See Also

RandOverRegress, RandUnderRegress

---

randOverRegress_ST *Biased over-sampling for imbalanced regression spatio-temporal problems*

Description

Based on randOverRegress (R package UBL). This function performs a random over-sampling strategy for imbalanced regression problems with a bias based on spatio-temporal contextual information. Basically a percentage of cases of the "class(es)" (bumps above a relevance threshold defined) selected by the user are randomly over-sampled with a sampling bias based on a spatio-temporal weight. Alternatively, it can either balance all the existing "classes" (the default) or it can "smoothly invert" the frequency of the examples in each class.

Usage

```
randOverRegress_ST(form, dat, alpha = 0.5, beta = 0.9, rel = "auto",
  thr.rel = 0.5, epsilon = 1e-04, C.perc = "balance", repl = TRUE,
  type = "add", site_id = "site_id", time = "time",
  sites_sf = NULL, lon = NULL, lat = NULL, crs = NULL)
```

Arguments

form	a model formula
dat	the original training set (with the unbalanced distribution)
alpha	weighting parameter for temporal and spatial re-sampling probabilities. Default 0.5

beta	weighting parameter for spatiotemporal weight and phi for re-sampling probabilities. Default 0.9
rel	relevance determined automatically (default) with uba package or provided by the user
thr.rel	relevance threshold above which a case is considered as belonging to the rare "class"
epsilon	minimum weight to be added to all observations. Default 1E-4
C.perc	A vector containing the over-sampling percentage/s to apply to all/each "class" (bump) obtained with the relevance threshold. Replicas of the examples are are randomly added in each "class". If only one percentage is provided this value is reused in all the "classes" that have values above the relevance threshold. A different percentage can be provided to each "class". In this case, the percentages should be provided in ascending order of target variable value. The over-sampling percentage(s), should be numbers above 0, meaning that the important cases (cases above the threshold) are over-sampled by the corresponding percentage. If the number 1 is provided then the number of extreme examples will be doubled. Alternatively, C.perc parameter may be set to "balance" or "extreme", cases where the over-sampling percentages are automatically estimated to either balance or invert the frequencies of the examples in the "classes" (bumps).
repl	allowed to perform sampling with replacement
type	character string indicating the type of bias used. Default is "add". More types to be added in future work
site_id	the name of the column containing location IDs
time	the column name of the time-stamp
sites_sf	An sf obejct containing station and IDs and geometry points of the locations. As an alternative, provide lon, lat, and crs
lon	the name of the column containing the location's longitude
lat	the name of the column containing the location's latitude
crs	the code for the Coordinate Reference System

Value

The function returns a data frame with the new data set resulting from the application of the spatiotemporally biased over-sampling strategy.

References

Paula Branco, Rita P. Ribeiro, Luis Torgo (2016)., UBL: an R Package for Utility-Based Learning, CoRR abs/1604.08079 [cs.MS], URL: http://arxiv.org/abs/1604.08079

See Also

RandOverRegress, sample_wts

RandUnderRegress *Random under-sampling for imbalanced regression problems*

Description

This function is identical to the function of the same name available on the R package UBL.
The function performs a random under-sampling strategy for imbalanced regression problems. Essentially, a percentage of cases of the "class(es)" (bumps below a relevance threshold defined) selected by the user are randomly removed. Alternatively, the strategy can be applied to either balance all the existing "classes"" or to "smoothly invert" the frequency of the examples in each "class".

Usage

```
RandUnderRegress(form, dat, rel = "auto", thr.rel = 0.5,
  C.perc = "balance", repl = FALSE)
```

Arguments

form	A formula describing the prediction problem.
dat	A data frame containing the original imbalanced data set.
rel	The relevance function which can be automatically ("auto") determined (the default) or may be provided by the user through a matrix with interpolating points.
thr.rel	A number indicating the relevance threshold below which a case is considered as belonging to the normal "class".
C.perc	A vector containing the under-sampling percentage/s to apply to all/each "class" (bump) obtained with the relevance threshold. Examples are randomly removed from the "class(es)". If only one percentage is provided this value is reused in all the "classes" that have values below the relevance threshold. A different percentage can be provided to each "class". In this case, the percentages should be provided in ascending order of target variable value. The under-sampling percentage(s), should be a number below 1, meaning that the normal cases (cases below the threshold) are under-sampled by the corresponding percentage. If the number 1 is provided then those examples are not changed. Alternatively, C.perc parameter may be set to "balance" or "extreme", cases where the under-sampling percentages are automatically estimated to either balance or invert the frequencies of the examples in the "classes" (bumps).
repl	A boolean value controlling the possibility of having repetition of examples in the under-sampled data set. Defaults to FALSE.

Details

The only difference between this function and the original function is in the requirements imposed on the argument C.perc.
This function performs a random under-sampling strategy for dealing with imbalanced regression problems. The examples removed are randomly selected among the examples belonging to the normal "class(es)" (bump of relevance below the threshold defined). The user can chose one or more bumps to be under-sampled.

Value

The function returns a data frame with the new data set resulting from the application of the random under-sampling strategy.

References

Paula Branco, Rita P. Ribeiro, Luis Torgo (2016)., UBL: an R Package for Utility-Based Learning, CoRR abs/1604.08079 [cs.MS], URL: http://arxiv.org/abs/1604.08079

See Also

RandUnderRegress, RandOverRegress

`randUnderRegress_ST` *Biased under-sampling for imbalanced regression spatio-temporal problems*

Description

Based on randUnderRegress (R package UBL). The function performs a random under-sampling strategy for imbalanced regression problems with a bias based on spatio-temporal contextual information. Essentially, a percentage of cases of the "class(es)" (bumps below a relevance threshold defined) selected by the user are randomly removed with a sampling bias based on a spatio-temporal weight. Alternatively, the strategy can be applied to either balance all the existing "classes"" or to "smoothly invert" the frequency of the examples in each "class".

Usage

```
randUnderRegress_ST(form, dat, alpha = 0.5, beta = 0.9, rel = "auto",
  thr.rel = 0.5, epsilon = 1e-04, C.perc = "balance", repl = FALSE,
  type = "add", site_id = "site_id", time = "time",
  sites_sf = NULL, lon = NULL, lat = NULL, crs = NULL)
```

Arguments

`form`	a model formula
`dat`	the original training set (with the unbalanced distribution)
`alpha`	weighting parameter for temporal and spatial re-sampling probabilities. Default 0.5
`beta`	weighting parameter for spatiotemporal weight and phi for re-sampling probabilities. Default 0.9
`rel`	relevance determined automatically (default) with uba package or provided by the user
`thr.rel`	relevance threshold above which a case is considered as belonging to the rare "class"

epsilon	minimum weight to be added to all observations. Default 1E-4
C.perc	A vector containing the over-sampling percentage/s to apply to all/each "class" (bump) obtained with the relevance threshold. Replicas of the examples are are randomly added in each "class". If only one percentage is provided this value is reused in all the "classes" that have values above the relevance threshold. A different percentage can be provided to each "class". In this case, the percentages should be provided in ascending order of target variable value. The over-sampling percentage(s), should be numbers above 0, meaning that the important cases (cases above the threshold) are over-sampled by the corresponding percentage. If the number 1 is provided then the number of extreme examples will be doubled. Alternatively, C.perc parameter may be set to "balance" or "extreme", cases where the over-sampling percentages are automatically estimated to either balance or invert the frequencies of the examples in the "classes" (bumps).
repl	allowed to perform sampling with replacement
type	character string indicating the type of bias used. Default is "add". More types to be added in future work
site_id	the name of the column containing location IDs
time	the column name of the time-stamp
sites_sf	An sf obejct containing station and IDs and geometry points of the locations. As an alternative, provide lon, lat, and crs
lon	the name of the column containing the location's longitude
lat	the name of the column containing the location's latitude
crs	the code for the Coordinate Reference System

Value

The function returns a data frame with the new data set resulting from the application of the spatio-temporally biased under-sampling strategy.

References

Paula Branco, Rita P. Ribeiro, Luis Torgo (2016)., UBL: an R Package for Utility-Based Learning, CoRR abs/1604.08079 [cs.MS], URL: http://arxiv.org/abs/1604.08079

See Also

RandUnderRegress, sample_wts

responseValues *Get response values of a dataset from a formula*

Description

Get response values of a dataset from a formula

Usage

```
responseValues(formula, data, na = NULL)
```

Arguments

formula	learning formula
data	data set to get the target values from
na	what action to perform if NAs are present. Default is na.fail

Value

A vector of the target values.

sample_wts *Get spatio-temporal re-sampling weights*

Description

A function that calculates different weights for re-sampling that is temporally and/or spatially biased.

Usage

```
sample_wts(form, df, phi.control, alpha = 0.5, beta = 0.9,
  rel.thr = 0.9, epsilon = 1e-04, site_id = "site_id",
  time = "time", sites_sf = NULL, lon = NULL, lat = NULL,
  crs = NULL)
```

Arguments

form	a formula describing the learning task
df	a data frame
phi.control	list of parameters as returned by function UBL::phi.control
alpha	weighting parameter for temporal and spatial re-sampling probabilities. Default 0.5

beta	weighting parameter for spatiotemporal weight and phi for re-sampling probabilities. Default 0.9
rel.thr	a relevance threshold above which an observation is considered relevant
epsilon	minimum weight to be added to all observations. Default 1E-4
site_id	the name of the column containing location IDs
time	the column name of the time-stamp
sites_sf	An sf obejct containing station and IDs and geometry points of the locations. As an alternative, provide lon, lat, and crs
lon	the name of the column containing the location's longitude
lat	the name of the column containing the location's latitude
crs	the code for the Coordinate Reference System

Details

phi gives the target variable's relevance (higher relevance: phi -> 1; lower relevance: phi -> 0); time_wts gives the observation's temporally biased re-sampling weight (most recent observations: w -> 1; oldest: w -> 0.); space_wts gives the observation's spatially biased re-sampling weight (farthest away from other relevant cases at time slice: d -> 1.). High time_wts or space_wts means the observation is more likely to be kept.

Value

a data.frame with relevance phi, temporally biased weights time_wts, and spatially biased weights space_wts for each row in df.

See Also

get_phi, get_time_wts, get_space_wts.

---

shuffle *Shuffle values/rows*

Description

Shuffle the values or rows of a vector or data frame

Usage

```
shuffle(x)
```

Arguments

x	a vector or data frame

Value

a vector or data frame

simple_workflow *A simple learning and prediction workflow*

Description

A simple learning and prediction workflow that may deal with NAs and use re-sampling techniques to balance an imbalanced regression problem.

Usage

```
simple_workflow(train, test, form, model = "lm", resample = NULL,
  resample.pars = NULL, handleNAs = NULL, min_train = 2,
  nORp = 0.2, time = "time", site_id = "site", ...)
```

Arguments

train	a data frame for training
test	a data frame for testing
form	a formula describing the model to learn
model	the name of the algorithm to use
resample	re-sampling technique to be used. Default is NULL.
resample.pars	parameters to be passed to re-sample function. Default is NULL.
handleNAs	string indicating how to deal with NAs. If "centralImput", training observations with at least 80% of non-NA columns, will have their NAs substituted by the mean value and testing observatiosn will have their NAs filled in with mean value regardless. Default is NULL.
min_train	a minimum number of observations that must be left to train a model. If there are not enough observations, predictions will be NA. Default is 2.
nORp	a maximum number or fraction of columns/rows with missing values above which a row/column will be removed from train before learning the model. Only works if handleNAs was set to centralImputation. Default is 0.2.
time	the name of the column in train and test containing time-stamps
site_id	the name of the column in train and test containing location IDs
...	other parameters to feed to model

Value

a data frame containing time-stamps, location IDs, true values and predicted values

summarize_metrics *Summarize metrics*

Description

Used for internal validation

Usage

```
summarize_metrics(int.res, metrics)
```

Arguments

int.res	A list with results obtained by running estimates
metrics	a list of metrics that should be summarized

Value

a named vector with median, IQR, mean, standard-deviation, and number of non-NA values of each metric

Tall_SPrand *Spatial CV*

Description

Fold allocation of k-fold CV using:

- all time
- shuffled individual locations

Usage

```
Tall_SPrand(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

Tblock_SPall *Temporally blocked CV*

Description

Fold allocation of k-fold CV using:

- blocked time
- all locations

Usage

```
Tblock_SPall(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

Tblock_SPrand *Temporal blocked and randomly assigned spatial CV*

Description

Fold allocation of k-fold CV using:

- blocked time
- randomly assigned locations

Usage

```
Tblock_SPrand(data, nfolds, t.nfolds = round(sqrt(nfolds)),
  sp.nfolds = round(sqrt(nfolds)), time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
t.nfolds	number of folds across time. Default is sqrt(nfolds)
sp.nfolds	number of folds across space. Default is sqrt(nfolds)
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

---

Trand_SPall *Temporal CV*

Description

Fold allocation of k-fold CV using:

- shuffled time
- all locations

Usage

```
Trand_SPall(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

`Trand_SPrand` *Classic k-fold CV*

Description

Fold allocation of classic k-fold CV:

- shuffled time
- shuffled locations

Usage

```
Trand_SPrand(data, nfolds, time = "time", site_id = "site")
```

Arguments

`data`	full dataset
`nfolds`	number of folds
`time`	column name of time-stamp in `data`. Default is "time"
`site_id`	column name of location identifier in `data`. Default is "site"

Value

A list with slots:

- `data`, possibly re-ordered
- `f`, a vector with the fold numbers (from 1 to `nfolds`) of each row in `data`

Index