Package 'mlim'

August 13, 2022

Type Package

Title Missing Data Imputation with Automated Machine Learning

Version 0.0.1

Depends R (>= 3.5.0)

Description Using automated machine learning, the package fine-tunes an Elastic Net or Gradient Boosting Machine model for imputing the missing observations of each variable. This procedure has been implemented for the first time by this package and is expected to outperform other packages for imputing missing data that do not fine-tune their models.

License MIT + file LICENSE

Encoding UTF-8

Imports h2o, VIM, missRanger, memuse, md.log

RoxygenNote 7.2.1

URL https://github.com/haghish/mlim,
https://www.sv.uio.no/psi/english/people/aca/haghish/

BugReports https://github.com/haghish/mlim/issues

NeedsCompilation no

Author E. F. Haghish [aut, cre, cph]

Maintainer E. F. Haghish <haghish@uio.no>

Repository CRAN

Date/Publication 2022-08-13 12:30:02 UTC

R topics documented:

mlim *missing data imputation with automated machine learning*

Description

imputes data.frame with mixed variable types using automated machine learning (AutoML)

Usage

```
mlim(
  data = NULL,
  load.mlim = NULL,
  algos = c("ELNET", "DRF"),
  preimpute = "rf",
  preimputed_df = NULL,
  ignore = NULL,
  init = TRUE,
  save.mlim = NULL,
  maxiter = 10L,
  miniter = 2L,
  cv = 10L,
  tuning_time = 180,
  max_models = NULL,
  matching = "AUTO",
  balance = NULL,
  ignore.rank = FALSE,
  weights_column = NULL,
  seed = NULL,
  verbosity = NULL,
  report = NULL,
  iteration_stopping_metric = "RMSE",
  iteration_stopping_tolerance = 0.005,
  stopping_metric = "AUTO",
  stopping_rounds = 3,
  stopping_tolerance = 0.001,
  cpu = -1,
  ram = NULL,
  flush = FALSE,
  shutdown = TRUE,
  sleep = 0.5,
  ...
)
```

Arguments

data	a data.frame or matrix with missing data to be imputed. if load.mlim is provided, this argument will be ignored.

`load.mlim` an object of class "mlim", which includes the data, arguments, and settings for re-running the imputation, from where it was previously stopped. the "mlim" object saves the current state of the imputation and is particularly recommended for large datasets or when the user specifies a computationally extensive settings (e.g. specifying several algorithms, increasing tuning time, etc.).

`algos` character. specify a vector of algorithms to be used in the process of auto-tuning. the supported main algorithms are `"ELNET"`, `"RF"`, `"GBM"`, `"DL"`, `"XGB"` (available for Mac and Linux), and `"Ensemble"`.

the default is `c("ELNET", "RF")`, which tunes fast.Note that the choice of algorithms to be trained can largely increase the runtime. for advice on algorithm selection visit https://github.com/haghish/mlim. GBM, DL, XGB, and Ensemble take the full given "tuning_time" (see below) to tune the best model for imputing he given variable. if `load.mlim` is provided, this argument will be ignored.

`preimpute` character. specifies the procedure for handling the missing data before initiating the procedures. the default procedure is "rf", which models the missing data with parallel Random Forest model. possible alternatives are `"knn"` or `"mm"`.

`preimputed_df` data.frame. if you have used another software for missing data imputation, you can still optimize the imputation by handing the data.frame to this argument, which will bypass the "preimpute" procedure.

`ignore` character vector of column names or index of columns that should should be ignored in the process of imputation.

`init` logical. should h2o Java server be initiated? the default is TRUE. however, if the Java server is already running, set this argument to FALSE.

`save.mlim` filename. if a filename is specified, an `mlim` object is saved after the end of each variable imputation. this object not only includes the imputed dataframe and estimated cross-validation error, but also includes the information needed for continuing the imputation, which is very useful feature for imputing large datasets, with a long runtime. this argument is activated by default and an mlim object is stored in the local directory named `"mlim.rds"`.

`maxiter` integer. maximum number of iterations. the default value is `10`, but it can be reduced to `3` (not recommended, see below).

`miniter` integer. minimum number of iterations. the default value is 2.

`cv` logical. specify number of k-fold Cross-Validation (CV). values of 10 or higher are recommended. default is 10.

`tuning_time` integer. maximum runtime (in seconds) for fine-tuning the imputation model for each variable in each iteration. the default time is 600 seconds but for a large dataset, you might need to provide a larger model development time. this argument also influences `max_models`, see below.

`max_models` integer. maximum number of models that can be generated in the proecess of fine-tuning the parameters. this value default to 100, meaning that for imputing each variable in each iteration, up to 100 models can be fine-tuned. increasing this value should be consistent with increasing `max_model_runtime_secs`, allowing the model to spend more time in the process of individualized fine-tuning. as a result, the better tuned the model, the more accurate the imputed values are expected to be

`matching` logical. if `TRUE`, imputed values are coerced to the closest value to the non-missing values of the variable. if set to "AUTO", 'mlim' decides whether to match or not, based on the variable classes. the default is "AUTO".

`balance` character vector, specifying variable names that should be balanced before imputation. balancing the prevalence might decrease the overall accuracy of the imputation, because it attempts to ensure the representation of the rare outcome. this argument is optional and intended for advanced users that impute a severely imbalance categorical (nominal) variable.

`ignore.rank` logical, if FALSE (default), ordinal variables are imputed as continuous integers with regression plus matching and are reverted to ordinal later again. this procedure is recommended. if FALSE, the rank of the categories will be ignored the the algorithm will try to optimize for classification accuracy. WARNING: the latter often results in very high classification accuracy but at the cost of higher rank error. see the "mlim.error" function documentation to see how rank error is computed. therefore, if you intend to carry out analysis on the rank data as numeric, it is recommended that you set this argument to FALSE.

`weights_column` non-negative integer. a vector of observation weights can be provided, which should be of the same length as the dataframe. giving an observation a weight of Zero is equivalent of ignoring that observation in the model. in contrast, a weight of 2 is equivalent of repeating that observation twice in the dataframe. the higher the weight, the more important an observation becomes in the modeling process. the default is NULL.

`seed` integer. specify the random generator seed

`verbosity` character. controls how much information is printed to console. the value can be "warn" (default), "info", "debug", or NULL.

`report` filename. if a filename is specified (e.g. report = "mlim.md"), the `"md.log"` R package is used to generate a Markdown progress report for the imputation. the format of the report is adopted based on the `'verbosity'` argument. the higher the verbosity, the more technical the report becomes. if verbosity equals "debug", then a log file is generated, which includes time stamp and shows the function that has generated the message. otherwise, a reduced markdown-like report is generated. default is NULL.

`iteration_stopping_metric`
character. specify the minimum improvement in the estimated error to proceed to the following iteration or stop the imputation. the default is 10^-4 for `"MAE"` (Mean Absolute Error). this criteria is only applied from the end of the fourth iteration.

`iteration_stopping_tolerance`
numeric. the minimum rate of improvement in estimated error metric to qualify the imputation for another round of iteration, if the `maxiter` is not yet reached. the default value is 50^-3, meaning that in each iteration, the error must be reduced by at least 0.5 iteration.

`stopping_metric`
character.

`stopping_rounds`
integer.

stopping_tolerance
: numeric.

cpu
: integer. number of CPUs to be dedicated for the imputation. the default takes all of the available CPUs.

ram
: integer. specifies the maximum size, in Gigabytes, of the memory allocation. large memory size is particularly advised, especially for multicore processes. the more you give the more you get!

flush
: logical (experimental). if TRUE, after each model, the server is cleaned to retrieve RAM. this feature is in testing mode.

shutdown
: logical. if TRUE, h2o server is closed after the imputation. the default is TRUE

sleep
: integer. number of seconds to wait after each interaction with h2o server. the default is 1 second. larger values might be needed depending on your computation power or dataset size.

...
: Arguments passed to `h2o.automl()`. The following arguments are e.g. incompatible with `ranger`: `write.forest`, `probability`, `split.select.weights`, `dependent.variable.name`, and `classification`.

Value

a `data.frame`, showing the estimated imputation error from the cross validation within the data.frame's attribution

Author(s)

E. F. Haghish

Examples

```
data(iris)
irisNA <- mlim.na(iris, p = 0.1, stratify = TRUE, seed = 2022)

# run the default imputation (fastest imputation via 'mlim')
MLIM <- mlim(irisNA)
mlim.error(MLIM, irisNA, iris)

# run GBM model and allow 15 minutes of tuning for each variable
MLIM <- mlim(irisNA, algos = "GBM", tuning_time=60*15)
mlim.error(MLIM, irisNA, iris)
```

`mlim.error` *imputation error*

Description

calculates NRMSE, missclassification rate, and miss-ranking absolute mean distance, scaled from 0 to 1, where 1 means maximum distance between the actual rank of a level with the imputed level.

Usage

```
mlim.error(imputed, incomplete, complete, varwise = FALSE, ignore.rank = FALSE)
```

Arguments

`imputed`	the imputed dataframe
`incomplete`	the dataframe with missing values
`complete`	the original dataframe with no missing values
`varwise`	logical, default is FALSE. if TRUE, in addition to mean accuracy for each variable type, the algorithm's performance for each variable (column) of the datast is also returned. if TRUE, instead of a numeric vector, a list is retuned.
`ignore.rank`	logical (default is FALSE, which is recommended). if TRUE, the accuracy of imputation of ordered factors (ordinal variables) will be evaluated based on 'missclassification rate' instead of normalized euclidean distance. this practice is not recommended because higher classification rate for ordinal variables does not guarantee lower distances between the imputed levels, despite the popularity of evaluating ordinal variables based on missclassification rate. in other words, assume an ordinal variable has 5 levels (1. strongly disagree, 2. disagree, 3. uncertain, 4. agree, 5.strongly agree). in this example, if "ignore.rank = TRUE", then an imputation that imputes level "5" as "4" is equally inaccurate as other algorithm that imputes level "5" as "1". therefore, if you have ordinal variables in your dataset, make sure you declare them as "ordered" factors to get the best imputation accuracy.

Value

a numeric vector is "varwise = FALSE", or otherwise a list

Author(s)

E. F. Haghish

Examples

```
data(iris)
```

```
# add 10% missing values, ensure missingness is stratified for factors
irisNA <- mlim.na(iris, p = 0.1, stratify = TRUE, seed = 2022)

# run the default imputation
MLIM <- mlim(irisNA)
mlim.error(MLIM, irisNA, iris)
```

---

mlim.na *syntaxProcessing*

Description

extracts performance metrics from cross-validation

Usage

```
mlim.na(x, p = 0.1, stratify = FALSE, classes = NULL, seed = NULL)
```

Arguments

x	data.frame. x must be strictly a data.frame and any other data.table classes will be rejected
p	percentage of missingness to be added to the data
stratify	logical. if TRUE (default), stratified sampling will be carried out, when adding NA values to 'factor' variables (either ordered or unordered). this feature makes evaluation of missing data imputation algorithms more fair, especially when the factor levels are imbalanced.
classes	character vector, specifying the variable classes that should be selected for adding NA values. the default value is NULL, meaning all variables will receive NA values with probability of 'p'. however, if you wish to add NA values only to a specific classes, e.g. 'numeric' variables or 'ordered' factors, specify them in this argument. e.g. write "classes = c('numeric', 'ordered')" if you wish to add NAs only to numeric and ordered factors.
seed	integer. a random seed number for reproducing the result (recommended)

Author(s)

E. F. Haghish

Examples

```
# adding stratified NA to an atomic vector
x <- as.factor(c(rep("M", 100), rep("F", 900)))
table(mlim.na(x, p=.5, stratify = TRUE))
```

```
# adding unstratified NAs to all variables of a data.frame
data(iris)
mlim.na(iris, p=0.5, stratify = FALSE, seed = 1)

# or add stratified NAs only to factor variables, ignoring other variables
mlim.na(iris, p=0.5, stratify = TRUE, classes = "factor", seed = 1)

# or add NAs to numeric variables
mlim.na(iris, p=0.5, classes = "numeric", seed = 1)
```

Index