Package: learningmachine (via r-universe)

September 20, 2024

Type Package

Title Machine Learning with Explanations and Uncertainty Quantification

Version 2.6.0

Date 2024-09-20

Author T. Moudiki

Maintainer T. Moudiki <thierry.moudiki@gmail.com>

Description Regression-based Machine Learning with explanations and uncertainty quantification.

License BSD_3_clause + file LICENSE

Imports Rcpp (>= 1.0.10), R6

Depends randtoolbox, tseries, memoise, foreach, skimr, snow, doSNOW

LinkingTo Rcpp

RoxygenNote 7.3.0

Encoding UTF-8

Suggests caret, e1071, ggplot2, glmnet, knitr, MASS, mlbench, palmerpenguins, pkgbuild, pROC, ranger, reshape2, rmarkdown, roxygen2, testthat (>= 3.0.0), xgboost

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://techtonique.r-universe.dev

RemoteUrl https://github.com/Techtonique/learningmachine

RemoteRef HEAD

RemoteSha 8fda36484935201a98ed07a355ff29972e5a2e2f

Contents

Base *‘Base‘ class*

Description

the ‘Base‘ class used by other objects; useful for extensions of the package, not for basic interactions with the package

Methods

Public methods:

- `Base$new()`
- `Base$get_name()`
- `Base$get_type()`
- `Base$get_model()`
- `Base$set_model()`
- `Base$get_method()`
- `Base$set_method()`
- `Base$get_pi_method()`
- `Base$set_pi_method()`
- `Base$get_level()`
- `Base$set_level()`
- `Base$get_B()`
- `Base$set_B()`
- `Base$get_nb_hidden()`
- `Base$set_nb_hidden()`
- `Base$get_nodes_sim()`
- `Base$set_nodes_sim()`
- `Base$get_activ()`
- `Base$set_activ()`
- `Base$set_engine()`
- `Base$get_engine()`
- `Base$get_params()`
- `Base$get_seed()`
- `Base$set_seed()`
- `Base$summary()`
- `Base$clone()`

Method `new()`**:** Create a new object.

Usage:

```
Base$new(
  name = "Base",
  type = "none",
  model = NULL,
  method = NULL,
  X_train = NULL,
  y_train = NULL,
  pi_method = c("none", "splitconformal", "kdesplitconformal", "bootsplitconformal",
   "jackknifeplus", "kdejackknifeplus", "bootjackknifeplus", "surrsplitconformal",
    "surrjackknifeplus"),
  level = 95,
  B = 100,
  nb_hidden = 0,
  nodes_sim = c("sobol", "halton", "unif"),
  activ = c("relu", "sigmoid", "tanh", "leakyrelu", "elu", "linear"),
  engine = NULL,
  params = NULL,
  seed = 123
)
```

Arguments:

`name` name of the class

`type` type of supervised learning method implemented

`model` fitted model

`method` supevised learning method

`X_train` training set features

`y_train` training set response

`pi_method` type of prediction interval in c("splitconformal", "kdesplitconformal", "bootsplitconformal", "jackknifeplus", "kdejackknifeplus", "bootjackknifeplus", "surrsplitconformal", "surrjackknifeplus")

`level` an integer; the level of confidence

`B` an integer; the number of simulations when `level` is not `NULL`

`nb_hidden` number of nodes in the hidden layer, for construction of a quasi- randomized network

`nodes_sim` type of 'simulations' for hidden nodes, if `nb_hidden` > 0; takes values in c("sobol", "halton", "unif")

`activ` activation function's name for the hidden layer, in the construction of a quasi-randomized network; takes values in c("relu", "sigmoid", "tanh", " leakyrelu", "elu", "linear")

`engine` contains fit and predict lower-level methods for the given `method`; do not modify by hand

`params` additional parameters passed to `method` when calling `fit`

`seed` an integer; reproducibility seed for methods that include randomization

Returns: A new 'Base' object.

Method `get_name()`**:**

Usage:

```
Base$get_name()
```

Method `get_type()`**:**

Usage:

```
Base$get_type()
```

Method `get_model()`**:**

Usage:

```
Base$get_model()
```

Method `set_model()`**:**

Usage:

```
Base$set_model(model)
```

Method `get_method()`**:**

Usage:

```
Base$get_method()
```

Method `set_method()`**:**

Usage:

```
Base$set_method(method)
```

Method `get_pi_method()`**:**

Usage:

```
Base$get_pi_method()
```

Method `set_pi_method()`**:**

Usage:

```
Base$set_pi_method(pi_method)
```

Method `get_level()`**:**

Usage:

```
Base$get_level()
```

Method `set_level()`**:**

Usage:

```
Base$set_level(level)
```

Method `get_B()`**:**

Usage:

```
Base$get_B()
```

Method `set_B()`**:**

Usage:

```
Base$set_B(B)
```

Method `get_nb_hidden()`**:**

Usage:

`Base$get_nb_hidden()`

Method `set_nb_hidden()`**:**

Usage:

`Base$set_nb_hidden(nb_hidden)`

Method `get_nodes_sim()`**:**

Usage:

`Base$get_nodes_sim()`

Method `set_nodes_sim()`**:**

Usage:

`Base$set_nodes_sim(nodes_sim)`

Method `get_activ()`**:**

Usage:

`Base$get_activ()`

Method `set_activ()`**:**

Usage:

`Base$set_activ(activ)`

Method `set_engine()`**:**

Usage:

`Base$set_engine(engine)`

Method `get_engine()`**:**

Usage:

`Base$get_engine()`

Method `get_params()`**:**

Usage:

`Base$get_params()`

Method `get_seed()`**:**

Usage:

`Base$get_seed()`

Method `set_seed()`**:**

Usage:

`Base$set_seed(seed)`

Method `summary()`**:**

Usage:

```
Base$summary(
  X,
  show_progress = TRUE,
  class_name = NULL,
  class_index = NULL,
  y = NULL,
  type_ci = c("student", "nonparametric", "bootstrap"),
  cl = NULL
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Base$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Classifier` *'Classifier' class*

Description

The 'Classifier' class contains supervised classification models

Details

This class implements models:

lm Linear model

bcn see https://www.researchgate.net/publication/380760578_Boosted_Configuration_neural_Networks_for_supervised_cl

extratrees Extremely Randomized Trees; see https://link.springer.com/article/10.1007/s10994-006-6226-1

glmnet Elastic Net Regression; see https://glmnet.stanford.edu/

krr Kernel Ridge Regression; see for example https://www.jstatsoft.org/article/view/v079i03(but the implementation is different)

ranger Random Forest; see https://www.jstatsoft.org/article/view/v077i01

ridge Ridge regression; see https://arxiv.org/pdf/1509.09169

xgboost a scalable tree boosting system see https://arxiv.org/abs/1603.02754

rvfl Random Vector Functional Network, see https://www.researchgate.net/publication/332292006_Online_Bayesian_Quasi Random_functional_link_networks_application_to_the_optimization_of_black_box_functions

Super class

`learningmachine::Base` -> `Classifier`

Public fields

`name` name of the class

`type` type of supervised learning method implemented

`model` fitted model

`method` supervised learning method in c('lm', 'ranger', 'extratrees', 'ridge', 'bcn', 'glmnet', 'krr', 'xgboost')

`X_train` training set features; do not modify by hand

`y_train` training set response; do not modify by hand

`pi_method` type of prediction set in c("splitconformal", "kdesplitconformal", "bootsplitconformal", "surrsplitconformal")

`level` an integer; the level of confidence (default is 95, for 95 per cent) for prediction sets

`type_prediction_set` a string; the type of prediction set (currently, only "score" method)

`B` an integer; the number of simulations when `level` is not `NULL`

`nb_hidden` number of nodes in the hidden layer, for construction of a quasi- randomized network

`nodes_sim` type of 'simulations' for hidden nodes, if `nb_hidden` > 0; takes values in c("sobol", "halton", "unif")

`activ` activation function's name for the hidden layer, in the construction of a quasi-randomized network; takes values in c("relu", "sigmoid", "tanh", " leakyrelu", "elu", "linear")

`engine` contains fit and predic lower-level methods for the given `method`; do not modify by hand

`params` additional parameters passed to `method` when calling `fit` do not modify by hand

`seed` an integer; reproducibility seed for methods that include randomization

Methods

Public methods:

- `Classifier$new()`
- `Classifier$get_type_prediction_set()`
- `Classifier$set_type_prediction_set()`
- `Classifier$fit()`
- `Classifier$predict_proba()`
- `Classifier$predict()`
- `Classifier$clone()`

Method `new()`**:** Create a new object.

Usage:

```
Classifier$new(
  name = "Classifier",
  type = "classification",
  model = NULL,
  method = NULL,
  X_train = NULL,
  y_train = NULL,
 pi_method = c("none", "kdesplitconformal", "bootsplitconformal", "surrsplitconformal"),
  level = 95,
  type_prediction_set = c("none", "score"),
  B = 100,
  nb_hidden = 0,
  nodes_sim = c("sobol", "halton", "unif"),
  activ = c("relu", "sigmoid", "tanh", "leakyrelu", "elu", "linear"),
  engine = NULL,
  params = NULL,
  seed = 123
)
```

Returns: A new ‘Classifier‘ object.

Method `get_type_prediction_set()`**:**

Usage:

```
Classifier$get_type_prediction_set()
```

Method `set_type_prediction_set()`**:**

Usage:

```
Classifier$set_type_prediction_set(type_prediction_set)
```

Method `fit()`**:** Fit model to training set

Usage:

```
Classifier$fit(X, y, ...)
```

Arguments:

`X` a matrix of covariates (i.e explanatory variables)

`y` a vector, the response (i.e variable to be explained)

`...` additional parameters to learning algorithm (see vignettes)

Method `predict_proba()`**:**

Usage:

```
Classifier$predict_proba(X)
```

Method `predict()`**:**

Usage:

```
Classifier$predict(X)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Classifier$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Regressor *'Regressor' class*

Description

The 'Regressor' class contains supervised regression models

Details

This class implements models:

lm Linear model

bcn see https://www.researchgate.net/publication/380760578_Boosted_Configuration_neural_Networks_for_supervised_cl

extratrees Extremely Randomized Trees; see https://link.springer.com/article/10.1007/s10994-006-6226-1

glmnet Elastic Net Regression; see https://glmnet.stanford.edu/

krr Kernel Ridge Regression; see for example https://www.jstatsoft.org/article/view/v079i03(but the implementation is different)

ranger Random Forest; see https://www.jstatsoft.org/article/view/v077i01

ridge Ridge regression; see https://arxiv.org/pdf/1509.09169

xgboost a scalable tree boosting system see https://arxiv.org/abs/1603.02754

svm Support Vector Machines, see https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf

rvfl Random Vector Functional Network, see https://www.researchgate.net/publication/332292006_Online_Bayesian_Quasi Random_functional_link_networks_application_to_the_optimization_of_black_box_functions

Super class

`learningmachine::Base` -> `Regressor`

Public fields

`name` name of the class

`type` type of supervised learning method implemented

`model` fitted model

`method` supervised learning method in c('lm', 'ranger', 'extratrees', 'ridge', 'bcn', 'glmnet', 'krr', 'xgboost', 'svm')

`X_train` training set features; do not modify by hand

`y_train` training set response; do not modify by hand

`pi_method` type of prediction interval in c("splitconformal", "kdesplitconformal", "bootsplitconformal", "jackknifeplus", "kdejackknifeplus", "bootjackknifeplus", "surrsplitconformal", "surrjackknifeplus")

`level` an integer; the level of confidence (default is 95, for 95 per cent) for prediction intervals

`B` an integer; the number of simulations when 'level' is not NULL

`nb_hidden` number of nodes in the hidden layer, for construction of a quasi- randomized network

`nodes_sim` type of 'simulations' for hidden nodes, if `nb_hidden` > 0; takes values in c("sobol", "halton", "unif")

`activ` activation function's name for the hidden layer, in the construction of a quasi-randomized network; takes values in c("relu", "sigmoid", "tanh", " leakyrelu", "elu", "linear")

`engine` contains fit and predic lower-level methods for the given `method`; do not modify by hand

`params` additional parameters passed to `method` when calling `fit` do not modify by hand

`seed` an integer; reproducibility seed for methods that include randomization

Methods

Public methods:

- `Regressor$new()`
- `Regressor$fit()`
- `Regressor$predict()`
- `Regressor$fit_predict()`
- `Regressor$update()`
- `Regressor$clone()`

Method `new()`**:** Create a new object.

Usage:

```
Regressor$new(
  name = "Regressor",
  type = "regression",
  model = NULL,
  method = NULL,
  X_train = NULL,
  y_train = NULL,
  pi_method = c("none", "splitconformal", "jackknifeplus", "kdesplitconformal",
    "bootsplitconformal", "kdejackknifeplus", "bootjackknifeplus", "surrsplitconformal",
    "surrjackknifeplus"),
  level = 95,
  B = 100,
  nb_hidden = 0,
  nodes_sim = c("sobol", "halton", "unif"),
  activ = c("relu", "sigmoid", "tanh", "leakyrelu", "elu", "linear"),
  engine = NULL,
  params = NULL,
  seed = 123
)
```

Returns: A new 'Regressor' object.

Method `fit()`**:** Fit model to training set

Usage:

```
Regressor$fit(X, y, type_split = c("stratify", "sequential"), ...)
```

Arguments:

`X` a matrix of covariates (i.e explanatory variables)

`y` a vector, the response (i.e variable to be explained)

`type_split` type of data splitting for split conformal prediction: "stratify" (for classical supervised learning) "sequential" (when the data sequential ordering matters)

`...` additional parameters to learning algorithm (see vignettes)

Method `predict()`**:** Predict model on test set

Usage:

```
Regressor$predict(X, ...)
```

Arguments:

`X` a matrix of covariates (i.e explanatory variables)

`...` additional parameters

Method `fit_predict()`**:** Fit model to training set and predict on test set

Usage:

```
Regressor$fit_predict(
  X,
  y,
  pct_train = 0.8,
  score = ifelse(is.factor(y), yes = function(preds, y_test) mean(preds == y_test), no =
    function(preds, y_test) sqrt(mean((preds - y_test)^2))),
  level = NULL,
  pi_method = c("none", "splitconformal", "jackknifeplus", "kdesplitconformal",
    "bootsplitconformal", "kdejackknifeplus", "bootjackknifeplus", "surrsplitconformal",
    "surrjackknifeplus"),
  B = 100,
  seed = 123,
  graph = FALSE,
  ...
)
```

Method `update()`**:** update model in an online fashion (for now, only implemented for 'rvfl' models")

Usage:

```
Regressor$update(newx, newy, ...)
```

Arguments:

`newx` a vector of new covariates (i.e explanatory variables)

`newy` a numeric, the new response's observation (i.e variable to be explained)

`...` additional parameters to be passed to the underlying model's method ‘update‘

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Regressor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index