# Package ‘stackgbm’

April 30, 2024

**Title**  Stacked Gradient Boosting Machines

**Version**  0.1.0

**Description**  A minimalist implementation of model stacking by Wolpert (1992) \(<\text{doi:10.1016/S0893-6080(05)80023-1}>\) for boosted tree models. A classic, two-layer stacking model is implemented, where the first layer generates features using gradient boosting trees, and the second layer employs a logistic regression model that uses these features as inputs. Utilities for training the base models and parameters tuning are provided, allowing users to experiment with different ensemble configurations easily. It aims to provide a simple and efficient way to combine multiple gradient boosting models to improve predictive model performance and robustness.

**License**  MIT + file LICENSE

**URL**  [https://nanx.me/stackgbm/](https://nanx.me/stackgbm/), [https://github.com/nanxstats/stackgbm](https://github.com/nanxstats/stackgbm)

**BugReports**  [https://github.com/nanxstats/stackgbm/issues](https://github.com/nanxstats/stackgbm/issues)

**Encoding**  UTF-8

**VignetteBuilder**  knitr

**Depends**  R (>= 3.5.0)

**Imports**  pROC, progress, rlang

**Suggests**  knitr, lightgbm, msaenet, rmarkdown, xgboost

**RoxygenNote**  7.3.1

**NeedsCompilation**  no

**Author**  Nan Xiao [aut, cre, cph] (<https://orcid.org/0000-0002-0250-5673>)

**Maintainer**  Nan Xiao &lt;me@nanx.me&gt;

**Repository**  CRAN

**Date/Publication**  2024-04-30 11:20:02 UTC
R topics documented:

- `catboost_load_pool` ........................................... 2
- `catboost_predict` ............................................ 3
- `catboost_train` .............................................. 4
- `cv_catboost` .................................................... 5
- `cv_lightgbm` .................................................... 6
- `cv_param_grid` .................................................. 8
- `cv_xgboost` ..................................................... 9
- `is_installed_catboost` ........................................ 10
- `is_installed_lightgbm` ........................................ 11
- `is_installed_xgboost` ......................................... 11
- `lightgbm_train` ................................................ 12
- `predict.stackgbm` ............................................. 13
- `stackgbm` ...................................................... 14
- `xgboost_dmatrix` ................................................ 15
- `xgboost_train` ................................................... 16

Index .............................................................. 18

catboost_load_pool  Create a dataset

Description

Create a dataset

Usage

catboost_load_pool(data, label = NULL, ...)

Arguments

data  Predictors.
label  Labels.
...  Additional parameters.

Value

A `catboost.Pool` object.
Examples

```
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

catboost_load_pool(data = sim_data$x.tr, label = sim_data$y.tr)
catboost_load_pool(data = sim_data$x.tr, label = NULL)
catboost_load_pool(data = sim_data$x.te, label = NULL)
```

**Description**

Predict based on the model

**Usage**

```
catboost_predict(model, pool, prediction_type = "Probability", ...)
```

**Arguments**

- model: The trained model.
- pool: The dataset to predict on.
- prediction_type: Prediction type.
- ...: Additional parameters.

**Value**

Predicted values.

**Examples**

```
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
```

```
snr = 1,
p.train = 0.8,
seed = 42
)

x_train <- catboost_load_pool(data = sim_data$x.tr, label = sim_data$y.tr)
x_test <- catboost_load_pool(data = sim_data$x.te, label = NULL)

fit <- catboost_train(x_train, NULL, params = list(loss_function = "Logloss", iterations = 100, depth = 3, logging_level = "Silent")
)

catboost_predict(fit, x_test)

---

**catboost_train**  
*Train the model*

**Description**  
Train the model

**Usage**  
```r
catboost_train(learn_pool, test_pool = NULL, params = list())
```

**Arguments**  
- `learn_pool`  
  Training dataset.
- `test_pool`  
  Testing dataset.
- `params`  
  A list of training parameters.

**Value**  
A model object.

**Examples**  
```r
sim_data <- msaenet::msaenet.sim.binomial(n = 100, p = 10,
```

cv_catboost

rho = 0.6,
coef = rnorm(5, mean = 0, sd = 10),
snr = 1,
p.train = 0.8,
seed = 42
)
x_train <- catboost_load_pool(data = sim_data$x.tr, label = sim_data$y.tr)

fit <- catboost_train(
  x_train,
  NULL,
  params = list(
    loss_function = "Logloss",
    iterations = 100,
    depth = 3,
    logging_level = "Silent"
  )
)

fit

cv_catboost

Description

catboost - parameter tuning and model selection with k-fold cross-validation and grid search

Usage

cv_catboost(
  x,
  y,
  params = cv_param_grid(),
  n_folds = 5,
  n_threads = 1,
  seed = 42,
  verbose = TRUE
)

Arguments

  x Predictor matrix.
  y Response vector.
  params Parameter grid generated by cv_param_grid().
cv_lightgbm

lightgbm - parameter tuning and model selection with k-fold cross-validation and grid search

**Description**

lightgbm - parameter tuning and model selection with k-fold cross-validation and grid search

n_folds  Number of folds. Default is 5.
n_threads The number of parallel threads. For optimal speed, match this to the number of physical CPU cores, not threads. See respective model documentation for more details. Default is 1.
seed Random seed for reproducibility.
verbose Show progress?

**Value**

A data frame containing the complete tuning grid and the AUC values, with the best parameter combination and the highest AUC value.

**Examples**

```r
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

params <- cv_catboost(
  sim_data$x.tr,
  sim_data$y.tr,
  params = cv_param_grid(
    n_iterations = c(100, 200),
    max_depth = c(3, 5),
    learning_rate = c(0.1, 0.5)
  ),
  n_folds = 5,
  n_threads = 1,
  seed = 42,
  verbose = FALSE
)

params$df
```
cv_lightgbm

Usage

```r
cv_lightgbm(
  x,
  y,
  params = cv_param_grid(),
  n_folds = 5,
  n_threads = 1,
  seed = 42,
  verbose = TRUE
)
```

Arguments

- **x**: Predictor matrix.
- **y**: Response vector.
- **params**: Parameter grid generated by `cv_param_grid()`.
- **n_folds**: Number of folds. Default is 5.
- **n_threads**: The number of parallel threads. For optimal speed, match this to the number of physical CPU cores, not threads. See respective model documentation for more details. Default is 1.
- **seed**: Random seed for reproducibility.
- **verbose**: Show progress?

Value

A data frame containing the complete tuning grid and the AUC values, with the best parameter combination and the highest AUC value.

Examples

```r
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

params <- suppressWarnings(
  cv_lightgbm(
    sim_data$x.tr,
    sim_data$y.tr,
    params = cv_param_grid(
      n_iterations = c(100, 200),
      max_depth = c(3, 5),
    
```
Generate a parameter grid for cross-validation

Description

This function generates a parameter grid to be used in the cross-validation of gradient boosting decision tree (GBDT) models.

Usage

```r
cv_param_grid(
  n_iterations = c(100, 200, 500, 1000),
  max_depth = c(3, 5, 7, 9),
  learning_rate = c(0.01, 0.05, 0.1, 0.2)
)
```

Arguments

- `n_iterations` A numeric vector of the number of iterations (trees) for the GBDT model. This is equivalent to `nrounds` in XGBoost, `num_iterations` in LightGBM, and `iterations` in CatBoost.
- `max_depth` A numeric vector of the maximum tree depths. This parameter is equivalent to `max_depth` in XGBoost and LightGBM, and `depth` in CatBoost.
- `learning_rate` A numeric vector of learning rates for the GBDT model. This parameter is equivalent to `eta` in XGBoost, `learning_rate` in LightGBM, and ignored in CatBoost.

Value

A list where the names are the parameter names and the values are vectors of possible values for those parameters.
Examples

```r
params <- cv_param_grid(
  n_iterations = c(10, 100),
  max_depth = c(3, 5),
  learning_rate = c(0.01, 0.1)
)
```

Description

xgboost - parameter tuning and model selection with k-fold cross-validation and grid search

Usage

```r
cv_xgboost(
  x,
  y,
  params = cv_param_grid(),
  n_folds = 5,
  n_threads = 1,
  seed = 42,
  verbose = TRUE
)
```

Arguments

- `x`: Predictor matrix.
- `y`: Response vector.
- `params`: Parameter grid generated by `cv_param_grid()`.
- `n_folds`: Number of folds. Default is 5.
- `n_threads`: The number of parallel threads. For optimal speed, match this to the number of physical CPU cores, not threads. See respective model documentation for more details. Default is 1.
- `seed`: Random seed for reproducibility.
- `verbose`: Show progress?

Value

A data frame containing the complete tuning grid and the AUC values, with the best parameter combination and the highest AUC value.
Examples

sim_data <- msaenet::msaenet.sim.binomial(n = 100,
p = 10,
rho = 0.6,
coef = rnorm(5, mean = 0, sd = 10),
snr = 1,
p.train = 0.8,
seed = 42)

params <- cv_xgboost(
sim_data$x.tr,
sim_data$y.tr,
params = cv_param_grid(
  n.iterations = c(100, 200),
  max.depth = c(3, 5),
  learning.rate = c(0.1, 0.5)
),
n.folds = 5,
n.threads = 1,
seed = 42,
verbose = FALSE
)

params$df

is_installed_catboost

Is catboost installed?

Description

Is catboost installed?

Usage

is_installed_catboost()

Value

TRUE if installed, FALSE if not.

Examples

is_installed_catboost()
is_installed_lightgbm  Is lightgbm installed?

Description
Is lightgbm installed?

Usage
is_installed_lightgbm()

Value
TRUE if installed, FALSE if not.

Examples
is_installed_lightgbm()

is_installed_xgboost  Is xgboost installed?

Description
Is xgboost installed?

Usage
is_installed_xgboost()

Value
TRUE if installed, FALSE if not.

Examples
is_installed_xgboost()
lightgbm_train  

# Train lightgbm model

## Description

Train lightgbm model

## Usage

lightgbm_train(data, label, params, ...)

## Arguments

- **data**: Training data.
- **label**: Labels.
- **params**: A list of parameters.
- **...**: Additional parameters.

## Value

A model object.

## Examples

```r
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

fit <- suppressWarnings(
  lightgbm_train(
    data = sim_data$x.tr,
    label = sim_data$y.tr,
    params = list(
      objective = "binary",
      learning_rate = 0.1,
      num_iterations = 100,
      max_depth = 3,
      num_leaves = 2^3 - 1,
      num_threads = 1
    ),
    verbose = -1
  )
)
```
predict.stackgbm

) )

fit

predict.stackgbm  Make predictions from a stackgbm model object

Description

Make predictions from a stackgbm model object

Usage

## S3 method for class 'stackgbm'
predict(object, newx, threshold = 0.5, classes = c(1L, 0L), ...)

Arguments

object  A stackgbm model object.
newx    New predictor matrix.
threshold Decision threshold. Default is 0.5.
classes The class encoding vector of the predicted outcome. The naming and order will be respected.
...     Unused.

Value

A list of two vectors presenting the predicted classification probabilities and predicted response.

Examples

sim_data <- msaenet::msaenet.sim.binomial(
  n = 1000,
  p = 50,
  rho = 0.6,
  coef = rnorm(25, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

params_xgboost <- structure(
  list("nrounds" = 200, "eta" = 0.05, "max_depth" = 3),
  class = c("cv_params", "cv_xgboost")
)
params_lightgbm <- structure(
  list("num_iterations" = 200, "max_depth" = 3, "learning_rate" = 0.05),
  class = c("cv_params", "cv_lightgbm")
)
params_catboost <- structure(
  list("iterations" = 100, "depth" = 3),
  class = c("cv_params", "cv_catboost")
)

fit <- stackgbm(
  sim_data$x.tr,
  sim_data$y.tr,
  params = list(
    params_xgboost,
    params_lightgbm,
    params_catboost
  )
)

predict(fit, newx = sim_data$x.te)

---

**stackgbm**

*Model stacking for boosted trees*

**Description**

Model stacking with a two-layer architecture: first layer being boosted tree models fitted by xgboost, lightgbm, and catboost; second layer being a logistic regression model.

**Usage**

```
stackgbm(x, y, params, n_folds = 5L, seed = 42, verbose = TRUE)
```

**Arguments**

- **x**
  - Predictor matrix.
- **y**
  - Response vector.
- **params**
  - A list of optimal parameter objects for boosted tree models derived from `cv_xgboost()`, `cv_lightgbm()`, and `cv_catboost()`. The order does not matter.
- **n_folds**
  - Number of folds. Default is 5.
- **seed**
  - Random seed for reproducibility.
- **verbose**
  - Show progress?

**Value**

Fitted boosted tree models and stacked tree model.
Examples

```r
sim_data <- msaenet::msaenet.sim.binomial(
  n = 1000,
  p = 50,
  rho = 0.6,
  coef = rnorm(25, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

params_xgboost <- structure(
  list("nrounds" = 200, "eta" = 0.05, "max_depth" = 3),
  class = c("cv_params", "cv_xgboost")
)
params_lightgbm <- structure(
  list("num_iterations" = 200, "max_depth" = 3, "learning_rate" = 0.05),
  class = c("cv_params", "cv_lightgbm")
)
params_catboost <- structure(
  list("iterations" = 100, "depth" = 3),
  class = c("cv_params", "cv_catboost")
)

fit <- stackgbm(
  sim_data$x.tr,
  sim_data$y.tr,
  params = list(
    params_xgboost,
    params_lightgbm,
    params_catboost
  )
)

predict(fit, newx = sim_data$x.te)
```

---

**xgboost_dmatrix**

*Create xgb.DMatrix object*

**Description**

Create xgb.DMatrix object

**Usage**

```r
xgboost_dmatrix(data, label = NULL, ...)
```
Arguments

- \texttt{data} \quad \text{Matrix or file.}
- \texttt{label} \quad \text{Labels (optional).}
- \ldots \quad \text{Additional parameters.}

Value

An \texttt{xgb.DMatrix} object.

Examples

```r
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)
xgboost_dmatrix(sim_data$x.tr, label = sim_data$y.tr)
xgboost_dmatrix(sim_data$x.te)
```

\texttt{xgboost\_train} \quad \textit{Train xgboost model}

Description

Train xgboost model

Usage

\texttt{xgboost\_train(params, data, nrounds, \ldots)}

Arguments

- \texttt{params} \quad \text{A list of parameters.}
- \texttt{data} \quad \text{Training data.}
- \texttt{nrounds} \quad \text{The Maximum number of boosting iterations.}
- \ldots \quad \text{Additional parameters.}

Value

A model object.
Examples

```r
sim_data <- msaenet::msaenet.sim.binomial(
  n = 100,
  p = 10,
  rho = 0.6,
  coef = rnorm(5, mean = 0, sd = 10),
  snr = 1,
  p.train = 0.8,
  seed = 42
)

x_train <- xgboost_dmatrix(sim_data$x.tr, label = sim_data$y.tr)

fit <- xgboost_train(
  params = list(
    objective = "binary:logistic",
    eval_metric = "auc",
    max_depth = 3,
    eta = 0.1
  ),
  data = x_train,
  nrounds = 100,
  nthread = 1
)

fit
```
Index

catboost_load_pool, 2
catboost_predict, 3
catboost_train, 4
cv_catboost, 5
cv_catboost(), 14
cv_lightgbm, 6
cv_lightgbm(), 14
cv_param_grid, 8
cv_param_grid(), 5, 7, 9
cv_xgboost, 9
cv_xgboost(), 14
is_installed_catboost, 10
is_installed_lightgbm, 11
is_installed_xgboost, 11
lightgbm_train, 12
predict.stackgbm, 13
stackgbm, 14
xgboost_dmatrix, 15
xgboost_train, 16