Package 'autoEnsemble'

July 22, 2025

Type Package

Title Automated Stacked Ensemble Classifier for Severe Class Imbalance

Version 0.3

Depends R (>= 3.5.0),

Description A stacking solution for modeling imbalanced and severely skewed data. It automates the process of building homogeneous or heterogeneous stacked ensemble models by selecting ``best'' models according to different criteria. In doing so, it strategically searches for and selects diverse, high-performing base-learners to construct ensemble models optimized for skewed data. This package is particularly useful for addressing class imbalance in datasets, ensuring robust and effective model outcomes through advanced ensemble strategies which aim to stabilize the model, reduce its overfitting, and further improve its generalizability.

License MIT + file LICENSE

Encoding UTF-8

Imports h2o (>= 3.34.0.0), h2otools (>= 0.3), curl (>= 4.3.0)

RoxygenNote 7.3.2

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

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

NeedsCompilation no

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

Maintainer E. F. Haghish <haghish@hotmail.com>

Repository CRAN

Date/Publication 2025-03-20 11:50:13 UTC

Contents

`autoEnsemble` *Automatically Trains H2O Models and Builds a Stacked Ensemble Model*

Description

Automatically trains various algorithms to build base-learners and then automatically creates a stacked ensemble model

Usage

```
autoEnsemble(
  x,
  y,
  training_frame,
  validation_frame = NULL,
  nfolds = 10,
  balance_classes = TRUE,
  max_runtime_secs = NULL,
  max_runtime_secs_per_model = NULL,
  max_models = NULL,
  sort_metric = "AUCPR",
  include_algos = c("GLM", "DeepLearning", "DRF", "XGBoost", "GBM"),
  save_models = FALSE,
  directory = paste("autoEnsemble", format(Sys.time(), "%d-%m-%y-%H:%M")),
  ...,
  newdata = NULL,
  family = "binary",
  strategy = c("search"),
  model_selection_criteria = c("auc", "aucpr", "mcc", "f2"),
  min_improvement = 1e-05,
  max = NULL,
  top_rank = seq(0.01, 0.99, 0.01),
  stop_rounds = 3,
  reset_stop_rounds = TRUE,
  stop_metric = "auc",
  seed = -1,
  verbatim = FALSE,
  startH2O = FALSE,
  nthreads = NULL,
  max_mem_size = NULL,
  min_mem_size = NULL,
```

```
  ignore_config = FALSE,
  bind_to_localhost = FALSE,
  insecure = TRUE
)
```

Arguments

x	Vector. Predictor column names or indices.
y	Character. The response column name or index.
training_frame	An H2OFrame containing the training data. Default is h2o.getFrame("hmda.train.hex").
validation_frame	An H2OFrame for early stopping. Default is NULL.
nfolds	Integer. Number of folds for cross-validation. Default is 10.
balance_classes	Logical. Specify whether to oversample the minority classes to balance the class distribution; only applicable to classification
max_runtime_secs	Integer. This argument specifies the maximum time that the AutoML process will run for in seconds.
max_runtime_secs_per_model	Maximum runtime in seconds dedicated to each individual model training process.
max_models	Maximum number of models to build in the AutoML training (passed to autoML)
sort_metric	Metric to sort the leaderboard by (passed to autoML). For binomial classification choose between "AUC", "AUCPR", "logloss", "mean_per_class_error", "RMSE", "MSE". For regression choose between "mean_residual_deviance", "RMSE", "MSE", "MAE", and "RMSLE". For multinomial classification choose between "mean_per_class_error", "logloss", "RMSE", "MSE". Default is "AUTO". If set to "AUTO", then "AUC" will be used for binomial classification, "mean_per_class_error" for multinomial classification, and "mean_residual_deviance" for regression.
include_algos	Vector of character strings naming the algorithms to restrict to during the model-building phase. this argument is passed to autoML.
save_models	Logical. if TRUE, the models trained will be stored locally
directory	path to a local directory to store the trained models
...	parameters to be passed to autoML algorithm in h2o package
newdata	h2o frame (data.frame). the data.frame must be already uploaded on h2o server (cloud). when specified, this dataset will be used for evaluating the models. if not specified, model performance on the training dataset will be reported.
family	model family. currently only "binary" classification models are supported.
strategy	character. the current available strategies are "search" (default) and "top". The "search" strategy searches for the best combination of top-performing diverse models whereas the "top" strategy is more simplified and just combines the specified of top-performing diverse models without examining the possibility of

improving the model by searching for larger number of models that can further improve the model. generally, the "search" strategy is preferable, unless the computation runtime is too large and optimization is not possible.

model_selection_criteria
: character, specifying the performance metrics that should be taken into consideration for model selection. the default are "c('auc', 'aucpr', 'mcc', 'f2')". other possible criteria are "'f1point5', 'f3', 'f4', 'f5', 'kappa', 'mean_per_class_error', 'gini', 'accuracy'", which are also provided by the "evaluate" function.

min_improvement
: numeric. specifies the minimum improvement in model evaluation metric to qualify further optimization search.

max
: integer. specifies maximum number of models for each criteria to be extracted. the default value is the "top_rank" percentage for each model selection criteria.

top_rank
: numeric vector. specifies percentage of the top models taht should be selected. if the strategy is "search", the algorithm searches for the best best combination of the models from top ranked models to the bottom. however, if the strategy is "top", only the first value of the vector is used (default value is top 1%).

stop_rounds
: integer. number of stoping rounds, in case the model stops improving

reset_stop_rounds
: logical. if TRUE, everytime the model improves the stopping rounds penalty is resets to 0.

stop_metric
: character. model stopping metric. the default is "auc", but "aucpr" and "mcc" are also available.

seed
: random seed (recommended)

verbatim
: logical. if TRUE, it reports additional information about the progress of the model training, particularly used for debugging.

startH2O
: Logical. if TRUE, h2o server will be initiated.

nthreads
: arguments to be passed to h2o.init()

max_mem_size
: arguments to be passed to h2o.init()

min_mem_size
: arguments to be passed to h2o.init()

ignore_config
: arguments to be passed to h2o.init()

bind_to_localhost
: arguments to be passed to h2o.init()

insecure
: arguments to be passed to h2o.init()

Value

a list including the ensemble model and the top-rank models that were used in the model

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)
library(autoEnsemble)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I tune 2 set of model grids and use them both
### for building the ensemble, just to set an example ...

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GLM, GBM, XGBoost, DRF, DeepLearning) for 120 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("DRF","GLM", "XGBoost", "GBM", "DeepLearning"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

#########################################################
### PREPARE H2O Grid (takes a couple of minutes)
#########################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                  hyper_params = list(ntrees = seq(1,50,1)),
                  grid_id = "ensemble_grid",

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

#########################################################
### PREPARE ENSEMBLE MODEL
#########################################################

### get the models' IDs from the AutoML and grid searches.
### this is all that is needed before building the ensemble,
### i.e., to specify the model IDs that should be evaluated.

ids    <- c(h2o.get_ids(aml), h2o.get_ids(grid))
top    <- ensemble(models = ids, training_frame = prostate, strategy = "top")
```

```
search <- ensemble(models = ids, training_frame = prostate, strategy = "search")

#########################################################
### EVALUATE THE MODELS
#########################################################
h2o.auc(aml@leader)                          # best model identified by h2o.automl
h2o.auc(h2o.getModel(grid@model_ids[[1]]))   # best model identified by grid search
h2o.auc(top$model).                          # ensemble model with 'top' search strategy
h2o.auc(search$model).                     # ensemble model with 'search' search strategy


## End(Not run)
```

`ensemble` *Builds Stacked Ensemble Model from H2O Models*

Description

Multiple trained H2O models are stacked to create an ensemble

Usage

```
ensemble(
  models,
  training_frame,
  newdata = NULL,
  family = "binary",
  strategy = c("search"),
  model_selection_criteria = c("auc", "aucpr", "mcc", "f2"),
  min_improvement = 1e-05,
  max = NULL,
  top_rank = seq(0.01, 0.99, 0.01),
  stop_rounds = 3,
  reset_stop_rounds = TRUE,
  stop_metric = "auc",
  seed = -1,
  verbatim = FALSE
)
```

Arguments

`models`	H2O search grid or AutoML grid or a character vector of H2O model IDs. the `"h2o.get_ids"` function from `"h2otools"` can retrieve the IDs from grids.
`training_frame`	h2o training frame (data.frame) for model training
`newdata`	h2o frame (data.frame). the data.frame must be already uploaded on h2o server (cloud). when specified, this dataset will be used for evaluating the models. if not specified, model performance on the training dataset will be reported.

`family` model family. currently only `"binary"` classification models are supported.

`strategy` character. the current available strategies are `"search"` (default) and `"top"`. The `"search"` strategy searches for the best combination of top-performing diverse models whereas the `"top"` strategy is more simplified and just combines the specified of top-performing diverse models without examining the possibility of improving the model by searching for larger number of models that can further improve the model. generally, the `"search"` strategy is preferable, unless the computation runtime is too large and optimization is not possible.

`model_selection_criteria`
character, specifying the performance metrics that should be taken into consideration for model selection. the default are `"c('auc', 'aucpr', 'mcc', 'f2')"`. other possible criteria are `"'f1point5', 'f3', 'f4', 'f5', 'kappa', 'mean_per_class_error', 'gini', 'accuracy'"`, which are also provided by the `"evaluate"` function.

`min_improvement`
numeric. specifies the minimum improvement in model evaluation metric to qualify further optimization search.

`max` integer. specifies maximum number of models for each criteria to be extracted. the default value is the `"top_rank"` percentage for each model selection criteria.

`top_rank` numeric vector. specifies percentage of the top models taht should be selected. if the strategy is `"search"`, the algorithm searches for the best best combination of the models from top ranked models to the bottom. however, if the strategy is `"top"`, only the first value of the vector is used (default value is top 1%).

`stop_rounds` integer. number of stoping rounds, in case the model stops improving

`reset_stop_rounds`
logical. if TRUE, every time the model improves the stopping rounds penalty is resets to 0.

`stop_metric` character. model stopping metric. the default is `"auc"`, but `"aucpr"` and `"mcc"` are also available.

`seed` random seed (recommended)

`verbatim` logical. if TRUE, it reports additional information about the progress of the model training, particularly used for debugging.

Value

a list including the ensemble model and the top-rank models that were used in the model

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)
library(autoEnsemble)
```

```
# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I tune 2 set of model grids and use them both
### for building the ensemble, just to set an example ...

########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
########################################################
# run AutoML to tune various models (GLM, GBM, XGBoost, DRF, DeepLearning) for 120 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("DRF","GLM", "XGBoost", "GBM", "DeepLearning"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

########################################################
### PREPARE H2O Grid (takes a couple of minutes)
########################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                  hyper_params = list(ntrees = seq(1,50,1)),
                  grid_id = "ensemble_grid",

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

########################################################
### PREPARE ENSEMBLE MODEL
########################################################

### get the models' IDs from the AutoML and grid searches.
### this is all that is needed before building the ensemble,
### i.e., to specify the model IDs that should be evaluated.

ids    <- c(h2o.get_ids(aml), h2o.get_ids(grid))
top    <- ensemble(models = ids, training_frame = prostate, strategy = "top")
search <- ensemble(models = ids, training_frame = prostate, strategy = "search")

########################################################
### EVALUATE THE MODELS
########################################################
h2o.auc(aml@leader)                          # best model identified by h2o.automl
h2o.auc(h2o.getModel(grid@model_ids[[1]]))   # best model identified by grid search
```

```
h2o.auc(top$model).                          # ensemble model with 'top' search strategy
h2o.auc(search$model).                     # ensemble model with 'search' search strategy


## End(Not run)
```

---

evaluate *Evaluate H2O Model(s) Performance*

Description

Multiple model performance metrics are computed for each model

Usage

```
evaluate(id, newdata = NULL, ...)
```

Arguments

id	a character vector of H2O model IDs retrieved from H2O Grid search or AutoML random search. the "h2o.get_ids" function from "h2otools" can retrieve the IDs from grids.
newdata	h2o frame (data.frame). the data.frame must be already uploaded on h2o server (cloud). when specified, this dataset will be used for evaluating the models. if not specified, model performance on the training dataset will be reported.
...	arguments to be passed to "h2o.performance" from H2O package

Value

a data.frame of various model performance metrics for each model

Author(s)

E. F. Haghish

Examples

```
## Not run:
library(h2o)
library(h2otools) #for h2o.get_ids() function
library(autoEnsemble)

# initiate the H2O server to train a grid of models
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# Run a grid search or AutoML search
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)
```

```
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 30,
                  seed = 2023, nfolds = 10, keep_cross_validation_predictions = TRUE)

# get the model IDs from the H2O Grid search or H2O AutoML Grid
ids <- h2otools::h2o.get_ids(aml)

# evaluate all the models and return a dataframe
evals <- evaluate(id = ids)

## End(Not run)
```

`h2o.get_ids` *h2o.get_ids*

Description

extracts the model IDs from H2O AutoML object or H2O grid

Usage

```
h2o.get_ids(automl)
```

Arguments

`automl`	a h2o `"AutoML"` grid object

Value

a character vector of trained models' names (IDs)

Author(s)

E. F. Haghish

Examples

```
## Not run:
library(h2o)
library(autoEnsemble)
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 30)

# get the model IDs
ids <- h2o.get_ids(aml)
```

```
## End(Not run)
```

`modelSelection` *Selects Diverse Top-Performing Models for Stacking an Ensemble Model*

Description

Multiple model performance metrics are computed

Usage

```
modelSelection(
  eval,
  family = "binary",
  top_rank = 0.01,
  max = NULL,
  model_selection_criteria = c("auc", "aucpr", "mcc", "f2")
)
```

Arguments

`eval`	an object of class `"ensemble.eval"` which is provided by 'evaluate' function. this object is a data.frame, including several performance metrics for the evaluated models.
`family`	model family. currently only `"binary"` classification models are supported.
`top_rank`	numeric. what percentage of the top model should be selected? the default value is top 1% models.
`max`	integer. specifies maximum number of models for each criteria to be extracted. the default value is the `"top_rank"` percentage for each model selection criteria.
`model_selection_criteria`	character, specifying the performance metrics that should be taken into consideration for model selection. the default are `"c('auc', 'aucpr', 'mcc', 'f2')"`. other possible criteria are `"'f1point5', 'f3', 'f4', 'f5', 'kappa', 'mean_per_class_error', 'gini', 'accuracy'"`, which are also provided by the `"evaluate"` function.

Value

a matrix of F-Measures for different thresholds or the highest F-Measure value

Author(s)

E. F. Haghish

Examples

```
## Not run:
library(h2o)
library(h2otools) #for h2o.get_ids() function
library(h2oEnsemble)

# initiate the H2O server to train a grid of models
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# Run a grid search or AutoML search
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 30,
                  seed = 2023, nfolds = 10, keep_cross_validation_predictions = TRUE)

# get the model IDs from the H2O Grid search or H2O AutoML Grid
ids <- h2otools::h2o.get_ids(aml)

# evaluate all the models and return a dataframe
evals <- evaluate(id = ids)

# perform model selection (up to top 10% of each criteria)
select <- modelSelection(eval = evals, top_rank = 0.1))

## End(Not run)
```

---

stopping_criteria *Stopping Criteria for Ending the Search*

Description

Defines criteria for ending the optimization search

Usage

```
stopping_criteria(
  df,
  round,
  stop,
  min_improvement,
  stop_rounds = 3,
  reset_stop_rounds = TRUE,
  stop_metric = "auc"
)
```

Arguments

df	data.frame. includes the metrics of ensemblem model performance
round	integer. the current round of optimization
stop	integer. current round of stopping penalty
min_improvement	numeric. specifies the minimum improvement in model evaluation metric to qualify further optimization search.
stop_rounds	integer. number of stoping rounds, in case the model stops improving
reset_stop_rounds	logical. if TRUE, everytime the model improves the stopping rounds penalty is resets to 0.
stop_metric	character. model stopping metric. the default is "auc", but "aucpr" and "mcc" are also available.

Value

a matrix of F-Measures for different thresholds or the highest F-Measure value

Author(s)

E. F. Haghish

Index