Package 'agua'

July 22, 2025

Title 'tidymodels' Integration with 'h2o'

Version 0.1.4

Description Create and evaluate models using 'tidymodels' and 'h2o' <https://h2o.ai/>. The package enables users to specify 'h2o' as an engine for several modeling methods.

License MIT + file LICENSE

URL https://agua.tidymodels.org/, https://github.com/tidymodels/agua

BugReports https://github.com/tidymodels/agua/issues

Depends parsnip

Imports cli, dials, dplyr, generics (>= 0.1.3), ggplot2, glue, h2o (>= 3.38.0.1), hardhat (>= 1.1.0), methods, pkgconfig, purrr, rlang, rsample, stats, tibble, tidyr, tune (>= 1.2.0), vctrs, workflows

Suggests covr, knitr, modeldata, recipes, rmarkdown, testthat (>= 3.0.0)

Config/Needs/website tidyverse/tidytemplate, doParallel, tidymodels, vip

Config/testthat/edition 3

Config/testthat/parallel false

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Max Kuhn [aut] (ORCID: <https://orcid.org/0000-0003-2402-136X>), Qiushi Yan [aut, cre], Steven Pawley [aut], Posit Software, PBC [cph, fnd]

Maintainer Qiushi Yan <qiushi.yann@gmail.com>

Repository CRAN

Date/Publication 2024-06-04 17:40:02 UTC

Contents

`agua_backend_options` *Control model tuning via* `h2o::h2o.grid()`

Description

Control model tuning via `h2o::h2o.grid()`

Usage

```
agua_backend_options(parallelism = 1)
```

Arguments

`parallelism`	Level of Parallelism during grid model building. 1 = sequential building (default). Use the value of 0 for adaptive parallelism - decided by H2O. Any number > 1 sets the exact number of models built in parallel.

`as_h2o` *Data conversion tools*

Description

Data conversion tools

Usage

```
as_h2o(df, destination_frame_prefix = "object")

## S3 method for class 'H2OFrame'
as_tibble(
  x,
  ...,
  .rows = NULL,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  rownames = pkgconfig::get_config("tibble::rownames", NULL)
)
```

Arguments

df	A R data frame.
destination_frame_prefix	A character string to use as the base name.
x	An H2OFrame.
...	Unused, for extensibility.
.rows	The number of rows, useful to create a 0-column tibble or just as an additional check.
.name_repair	Treatment of problematic column names: • "minimal": No name repair or checks, beyond basic existence, • "unique": Make sure names are unique and not empty, • "check_unique": (default value), no name repair, but check they are unique, • "universal": Make the names unique and syntactic • a function: apply custom name repair (e.g., .name_repair = make.names for names in the style of base R). • A purrr-style anonymous function, see rlang::as_function() This argument is passed on as repair to vctrs::vec_as_names(). See there for more details on these terms and the strategies used to enforce them.
rownames	How to treat existing row names of a data frame or matrix: • NULL: remove row names. This is the default. • NA: keep row names. • A string: the name of a new column. Existing rownames are transferred into this column and the row.names attribute is deleted. No name repair is applied to the new column name, even if x already contains a column of that name. Use as_tibble(rownames_to_column(...)) to safeguard against this case. Read more in rownames.

Value

A tibble or, for as_h2o(), a list with data (an H2OFrame) and id (the id on the h2o server).

Examples

```
# start with h2o::h2o.init()
if (h2o_running()) {
  cars2 <- as_h2o(mtcars)
  cars2
  class(cars2$data)

  cars0 <- as_tibble(cars2$data)
  cars0
}
```

`autoplot.workflow` *Plot rankings and metrics of H2O AutoML results*

Description

The `autoplot()` method plots cross validation performances of candidate models in H2O AutoML output via facets on each metric.

Usage

```
## S3 method for class 'workflow'
autoplot(object, ...)

## S3 method for class 'H2OAutoML'
autoplot(
  object,
  type = c("rank", "metric"),
  metric = NULL,
  std_errs = qnorm(0.95),
  ...
)
```

Arguments

`object`	A fitted `auto_ml()` model.
`...`	Other options to pass to `autoplot()`.
`type`	A character value for whether to plot average ranking ("rank") or metrics ("metric").
`metric`	A character vector or NULL for which metric to plot. By default, all metrics will be shown via facets.
`std_errs`	The number of standard errors to plot.

Value

A ggplot object.

Examples

```
if (h2o_running()) {
  auto_fit <- auto_ml() %>%
    set_engine("h2o", max_runtime_secs = 5) %>%
    set_mode("regression") %>%
    fit(mpg ~ ., data = mtcars)

  autoplot(auto_fit)
}
```

h2o_predict *Prediction wrappers for h2o*

Description

Prediction wrappers for fitted models with h2o engine that include data conversion, h2o server cleanup, and so on.

Usage

```
h2o_predict(object, new_data, ...)

h2o_predict_classification(object, new_data, type = "class", ...)

h2o_predict_regression(object, new_data, type = "numeric", ...)

## S3 method for class '`_H2OAutoML`'
predict(object, new_data, id = NULL, ...)
```

Arguments

object	An object of class model_fit.
new_data	A rectangular data object, such as a data frame.
...	Other options passed to h2o::h2o.predict()
type	A single character value or NULL. Possible values are "numeric", "class", "prob", "conf_int", "pred_int", "quantile", "time", "hazard", "survival", or "raw". When NULL, predict() will choose an appropriate value based on the model's mode.
id	Model id in AutoML results.

Details

For AutoML, prediction is based on the best performing model.

Value

For type != "raw", a prediction data frame with the same number of rows as new_data. For type == "raw", return the result of h2o::h2o.predict().

Examples

```
if (h2o_running()) {
  spec <-
    rand_forest(mtry = 3, trees = 100) %>%
    set_engine("h2o") %>%
    set_mode("regression")
```

```
  set.seed(1)
  mod <- fit(spec, mpg ~ ., data = mtcars)
  h2o_predict_regression(mod$fit, new_data = head(mtcars), type = "numeric")

  # using parsnip
  predict(mod, new_data = head(mtcars))
}
```

---

h2o_start *Utility functions for interacting with the h2o server*

Description

Utility functions for interacting with the h2o server

Usage

```
h2o_start()

h2o_end()

h2o_running(verbose = FALSE)

h2o_remove(id)

h2o_remove_all()

h2o_get_model(id)

h2o_get_frame(id)

h2o_xgboost_available()
```

Arguments

verbose	Print out the message if no cluster is available.
id	Model or frame id.

Examples

```
## Not run:
if (!h2o_running()) {
  h2o_start()
}

## End(Not run)
```

h2o_train *Model wrappers for h2o*

Description

Basic model wrappers for h2o model functions that include data conversion, seed configuration, and so on.

Usage

```
h2o_train(
  x,
  y,
  model,
  weights = NULL,
  validation = NULL,
  save_data = FALSE,
  ...
)

h2o_train_rf(x, y, ntrees = 50, mtries = -1, min_rows = 1, ...)

h2o_train_xgboost(
  x,
  y,
  ntrees = 50,
  max_depth = 6,
  min_rows = 1,
  learn_rate = 0.3,
  sample_rate = 1,
  col_sample_rate = 1,
  min_split_improvement = 0,
  stopping_rounds = 0,
  validation = NULL,
  ...
)

h2o_train_gbm(
  x,
  y,
  ntrees = 50,
  max_depth = 6,
  min_rows = 1,
  learn_rate = 0.3,
  sample_rate = 1,
  col_sample_rate = 1,
  min_split_improvement = 0,
```

```
  stopping_rounds = 0,
  ...
)

h2o_train_glm(x, y, lambda = NULL, alpha = NULL, ...)

h2o_train_nb(x, y, laplace = 0, ...)

h2o_train_mlp(
  x,
  y,
  hidden = 200,
  l2 = 0,
  hidden_dropout_ratios = 0,
  epochs = 10,
  activation = "Rectifier",
  validation = NULL,
  ...
)

h2o_train_rule(
  x,
  y,
  rule_generation_ntrees = 50,
  max_rule_length = 5,
  lambda = NULL,
  ...
)

h2o_train_auto(x, y, verbosity = NULL, save_data = FALSE, ...)
```

Arguments

x	A data frame of predictors.
y	A vector of outcomes.
model	A character string for the model. Current selections are "automl", "randomForest", "xgboost", "gbm", "glm", "deeplearning", "rulefit" and "naiveBayes". Use h2o_xgboost_available() to see if xgboost can be used on your OS/h2o server.
weights	A numeric vector of case weights.
validation	An integer between 0 and 1 specifying the *proportion* of the data reserved as validation set. This is used by h2o for performance assessment and potential early stopping. Default to 0.
save_data	A logical for whether training data should be saved on the h2o server, set this to TRUE for AutoML models that needs to be re-fitted.
...	Other options to pass to the h2o model functions (e.g., h2o::h2o.randomForest()).
ntrees	Number of trees. Defaults to 50.

`mtries` Number of variables randomly sampled as candidates at each split. If set to -1, defaults to sqrt{p} for classification and p/3 for regression (where p is the # of predictors Defaults to -1.

`min_rows` Fewest allowed (weighted) observations in a leaf. Defaults to 1.

`max_depth` Maximum tree depth (0 for unlimited). Defaults to 20.

`learn_rate` (same as eta) Learning rate (from 0.0 to 1.0) Defaults to 0.3.

`sample_rate` Row sample rate per tree (from 0.0 to 1.0) Defaults to 0.632.

`col_sample_rate` (same as colsample_bylevel) Column sample rate (from 0.0 to 1.0) Defaults to 1.

`min_split_improvement` Minimum relative improvement in squared error reduction for a split to happen Defaults to 1e-05.

`stopping_rounds` Early stopping based on convergence of stopping_metric. Stop if simple moving average of length k of the stopping_metric does not improve for k:=stopping_rounds scoring events (0 to disable) Defaults to 0.

`lambda` Regularization strength

`alpha` Distribution of regularization between the L1 (Lasso) and L2 (Ridge) penalties. A value of 1 for alpha represents Lasso regression, a value of 0 produces Ridge regression, and anything in between specifies the amount of mixing between the two. Default value of alpha is 0 when SOLVER = 'L-BFGS'; 0.5 otherwise.

`laplace` Laplace smoothing parameter Defaults to 0.

`hidden` Hidden layer sizes (e.g. [100, 100]). Defaults to c(200, 200).

`l2` L2 regularization (can add stability and improve generalization, causes many weights to be small. Defaults to 0.

`hidden_dropout_ratios` Hidden layer dropout ratios (can improve generalization), specify one value per hidden layer, defaults to 0.5.

`epochs` How many times the dataset should be iterated (streamed), can be fractional. Defaults to 10.

`activation` Activation function. Must be one of: "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout". Defaults to Rectifier.

`rule_generation_ntrees` Specifies the number of trees to build in the tree model. Defaults to 50. Defaults to 50.

`max_rule_length` Maximum length of rules. Defaults to 3.

`verbosity` Verbosity of the backend messages printed during training; Must be one of NULL (live log disabled), "debug", "info", "warn", "error". Defaults to NULL.

Value

An h2o model object.

Examples

```
# start with h2o::h2o.init()
if (h2o_running()) {
 # ------------------------------------------------------------------------
 # Using the model wrappers:
 h2o_train_glm(mtcars[, -1], mtcars$mpg)

 # ------------------------------------------------------------------------
 # using parsnip:

 spec <-
   rand_forest(mtry = 3, trees = 500) %>%
   set_engine("h2o") %>%
   set_mode("regression")

 set.seed(1)
 mod <- fit(spec, mpg ~ ., data = mtcars)
 mod

 predict(mod, head(mtcars))
}
```

`rank_results.workflow` *Tools for working with H2O AutoML results*

Description

Functions that returns a tibble describing model performances.

- `rank_results()` ranks average cross validation performances of candidate models on each metric.
- `collect_metrics()` computes average statistics of performance metrics (summarized) for each model, or raw value in each resample (unsummarized).
- `tidy()` computes average performance for each model.
- `member_weights()` computes member importance for stacked ensemble models, i.e., the relative importance of base models in the meta-learner. This is typically the coefficient magnitude in the second-level GLM model.

`extract_fit_engine()` extracts single candidate model from `auto_ml()` results. When `id` is null, it returns the leader model.

`refit()` re-fits an existing AutoML model to add more candidates. The model to be re-fitted needs to have engine argument `save_data = TRUE`, and `keep_cross_validation_predictions = TRUE` if stacked ensembles is needed for later models.

Usage

```
## S3 method for class 'workflow'
rank_results(x, ...)

## S3 method for class '`_H2OAutoML`'
rank_results(x, ...)

## S3 method for class 'H2OAutoML'
rank_results(x, n = NULL, id = NULL, ...)

## S3 method for class 'workflow'
collect_metrics(x, ...)

## S3 method for class '`_H2OAutoML`'
collect_metrics(x, ...)

## S3 method for class 'H2OAutoML'
collect_metrics(x, summarize = TRUE, n = NULL, id = NULL, ...)

## S3 method for class '`_H2OAutoML`'
tidy(x, n = NULL, id = NULL, keep_model = TRUE, ...)

get_leaderboard(x, n = NULL, id = NULL)

member_weights(x, ...)

## S3 method for class '`_H2OAutoML`'
extract_fit_parsnip(x, id = NULL, ...)

## S3 method for class '`_H2OAutoML`'
extract_fit_engine(x, id = NULL, ...)

## S3 method for class 'workflow'
refit(object, ...)

## S3 method for class '`_H2OAutoML`'
refit(object, verbosity = NULL, ...)
```

Arguments

`...`	Not used.
`n`	An integer for the number of top models to extract from AutoML results, default to all.
`id`	A character vector of model ids to retrieve.
`summarize`	A logical; should metrics be summarized over resamples (TRUE) or return the values for each individual resample.

keep_model	A logical value for if the actual model object should be retrieved from the server. Defaults to TRUE.
object, x	A fitted auto_ml() model or workflow.
verbosity	Verbosity of the backend messages printed during training; Must be one of NULL (live log disabled), "debug", "info", "warn", "error". Defaults to NULL.

Details

H2O associates with each model in AutoML an unique id. This can be used for model extraction and prediction, i.e., extract_fit_engine(x, id = id) returns the model and predict(x, id = id) will predict for that model. extract_fit_parsnip(x, id = id) wraps the h2o model with parsnip parsnip model object is discouraged.

The algorithm column corresponds to the model family H2O use for a particular model, including xgboost ("XGBOOST"), gradient boosting ("GBM"), random forest and variants ("DRF", "XRT"), generalized linear model ("GLM"), and neural network ("deeplearning"). See the details section in h2o::h2o.automl() for more information.

Value

A tibble::tibble().

Examples

```
if (h2o_running()) {
 auto_fit <- auto_ml() %>%
   set_engine("h2o", max_runtime_secs = 5) %>%
   set_mode("regression") %>%
   fit(mpg ~ ., data = mtcars)

   rank_results(auto_fit, n = 5)
   collect_metrics(auto_fit, summarize = FALSE)
   tidy(auto_fit)
   member_weights(auto_fit)
}
```

Index