Package 'ebm'

May 8, 2026

Type Package

Title Explainable Boosting Machines

Version 0.1.0

Description An interface to the 'Python' 'InterpretML' framework for fitting explainable boosting machines (EBMs); see Nori et al. (2019) <doi:10.48550/arXiv.1909.09223> for details. EBMs are a modern type of generalized additive model that use tree-based, cyclic gradient boosting with automatic interaction detection. They are often as accurate as state-of-the-art blackbox models while remaining completely interpretable.

Depends R (>= 3.5.0)

Imports reticulate, ggplot2 (>= 0.9.0), lattice

Suggests htmltools, ISLR2, knitr, rmarkdown, rstudioapi

URL https://github.com/bgreenwell/ebm,
https://bgreenwell.github.io/ebm/

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Brandon M. Greenwell [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8120-0084>)

Maintainer Brandon M. Greenwell <greenwell.brandon@gmail.com>

Repository CRAN

Date/Publication 2025-03-05 15:50:08 UTC

Contents

`as.ebm` *Coerce to an EBM object*

Description

If possible, coerces its input to an ebm object.

Usage

```
as.ebm(x, ...)
```

Arguments

`x`	An object that inherits from class "`interpret.glassbox._ebm._ebm.ExplainableBoostingClassifi` or "`interpret.glassbox._ebm._ebm.ExplainableBoostingRegressor`". For instance, the result of calling `fit$copy()` where `fit` is a fitted `ebm()` object.
`...`	Additional optional arguments. (Currently ignored.)

Value

An ebm object that can be used with the associated methods `plot()` and so forth.

`ebm` *Explainable Boosting Machine (EBM)*

Description

This function is an R wrapper for the explainable boosting functions in the Python interpret library. It trains an Explainable Boosting Machine (EBM) model, which is a tree-based, cyclic gradient boosting generalized additive model with automatic interaction detection. EBMs are often as accurate as state-of-the-art blackbox models while remaining completely interpretable.

Usage

```
ebm(
  formula,
  data,
  max_bins = 1024L,
  max_interaction_bins = 64L,
  interactions = 0.9,
  exclude = NULL,
  validation_size = 0.15,
  outer_bags = 16L,
  inner_bags = 0L,
  learning_rate = 0.04,
  greedy_ratio = 10,
  cyclic_progress = FALSE,
  smoothing_rounds = 500L,
  interaction_smoothing_rounds = 100L,
  max_rounds = 25000L,
  early_stopping_rounds = 100L,
  early_stopping_tolerance = 1e-05,
  min_samples_leaf = 4L,
  min_hessian = 0,
  reg_alpha = 0,
  reg_lambda = 0,
  max_delta_step = 0,
  gain_scale = 5,
  min_cat_samples = 10L,
  cat_smooth = 10,
  missing = "separate",
  max_leaves = 2L,
  monotone_constraints = NULL,
  objective = c("auto", "log_loss", "rmse", "poisson_deviance",
   "tweedie_deviance:variance_power=1.5", "gamma_deviance", "pseudo_huber:delta=1.0",
    "rmse_log"),
  n_jobs = -1L,
  random_state = 42L,
  ...
)
```

Arguments

formula	A formula of the form y ~ x1 + x2 + ...
data	A data frame containing the variables in the model.
max_bins	Max number of bins per feature for the main effects stage. Default is 1024.
max_interaction_bins	Max number of bins per feature for interaction terms. Default is 64.
interactions	Interaction terms to be included in the model. Default is 0.9. Current options include:

- Integer (1 <= interactions): Count of interactions to be automatically selected.
- Percentage (interactions < 1.0): Determine the integer count of interactions by multiplying the number of features by this percentage.
- List of numeric pairs: The pairs contain the indices of the features within each additive term. In addition to pairs, the interactions parameter accepts higher order interactions. It also accepts univariate terms which will cause the algorithm to boost the main terms at the same time as the interactions. When boosting mains at the same time as interactions, the `exclude` parameter should be set to `"mains"` and currently `max_bins` needs to be equal to `max_interaction_bins`.

`exclude` Features or terms to be excluded. Default is `NULL`.

`validation_size`

Validation set size. Used for early stopping during boosting, and is needed to create outer bags. Default is 0.15. Options are:

- Integer (1 <= `validation_size`): Count of samples to put in the validation sets.
- Percentage (`validation_size` < 1.0): Percentage of the data to put in the validation sets.
- 0: Turns off early stopping. Outer bags have no utility. Error bounds will

`outer_bags` Number of outer bags. Outer bags are used to generate error bounds and help with smoothing the graphs.

`inner_bags` Number of inner bags. Default is 0 which turns off inner bagging.

`learning_rate` Learning rate for boosting. Deafult is 0.04.

`greedy_ratio` The proportion of greedy boosting steps relative to cyclic boosting steps. A value of 0 disables greedy boosting, effectively turning it off. Default is 10.

`cyclic_progress`

This parameter specifies the proportion of the boosting cycles that will actively contribute to improving the model's performance. It is expressed as a logical or numeric between 0 and 1, with the default set to `TRUE` (1.0), meaning 100% of the cycles are expected to make forward progress. If forward progress is not achieved during a cycle, that cycle will not be wasted; instead, it will be used to update internal gain calculations related to how effective each feature is in predicting the target variable. Setting this parameter to a value less than 1.0 can be useful for preventing overfitting. Default is `FALSE`.

`smoothing_rounds`

Number of initial highly regularized rounds to set the basic shape of the main effect feature graphs. Default is 500.

`interaction_smoothing_rounds`

Number of initial highly regularized rounds to set the basic shape of the interaction effect feature graphs during fitting. Default is 100.

`max_rounds` Total number of boosting rounds with `n_terms` boosting steps per round. Default is 25000.

`early_stopping_rounds`

Number of rounds with no improvement to trigger early stopping. 0 turns off early stopping and boosting will occur for exactly `max_rounds`. Default is 100.

early_stopping_tolerance
: Tolerance that dictates the smallest delta required to be considered an improvement which prevents the algorithm from early stopping. early_stopping_tolerance is expressed as a percentage of the early stopping metric. Negative values indicate that the individual models should be overfit before stopping. EBMs are a bagged ensemble of models. Setting the early_stopping_tolerance to zero (or even negative), allows learning to overfit each of the individual models a little, which can improve the accuracy of the ensemble as a whole. Overfitting each of the individual models reduces the bias of each model at the expense of increasing the variance (due to overfitting) of the individual models. But averaging the models in the ensemble reduces variance without much change in bias. Since the goal is to find the optimum bias-variance tradeoff for the ensemble of models—not the individual models—a small amount of overfitting of the individual models can improve the accuracy of the ensemble as a whole. Default is 1e-05.

min_samples_leaf
: Minimum number of samples allowed in the leaves. Default is 4.

min_hessian
: Minimum hessian required to consider a potential split valid. Default is 0.0.

reg_alpha
: L1 regularization. Default is 0.0.

reg_lambda
: L2 regularization. Default is 0.0.

max_delta_step
: Used to limit the max output of tree leaves; <=0.0 means no constraint. Default is 0.0.

gain_scale
: Scale factor to apply to nominal categoricals. A scale factor above 1.0 will cause the algorithm focus more on the nominal categoricals. Default is 5.0.

min_cat_samples
: Minimum number of samples in order to treat a category separately. If lower than this threshold the category is combined with other categories that have low numbers of samples. Default is 10.

cat_smooth
: Used for the categorical features. This can reduce the effect of noises in categorical features, especially for categories with limited data. Default is 10.0.

missing
: Method for handling missing values during boosting. Default is "separate". The placement of the missing value bin can influence the resulting model graphs. For example, placing the bin on the "low" side may cause missing values to affect lower bins, and vice versa. This parameter does not affect the final placement of the missing bin in the model (the missing bin will remain at index 0 in the term_scores_ attribute). Possible values for missing are:

  - "low": Place the missing bin on the left side of the graphs.
  - "high": Place the missing bin on the right side of the graphs.
  - "separate": Place the missing bin in its own leaf during each boosting step, effectively making it location-agnostic. This can lead to overfitting, especially when the proportion of missing values is small.
  - "gain": Choose the best leaf for the missing value contribution at each boosting step, based on gain.

max_leaves
: Maximum number of leaves allowed in each tree. Default is 2.

`monotone_constraints`
: Default is NULL. This parameter allows you to specify monotonic constraints for each feature's relationship with the target variable during model fitting. However, it is generally recommended to apply monotonic constraints post-fit using the `monotonize()` attribute rather than setting them during the fitting process. This recommendation is based on the observation that, during fitting, the boosting algorithm may compensate for a monotone constraint on one feature by utilizing another correlated feature, potentially obscuring any monotonic violations. If you choose to define monotone constraints, `monotone_constraints` should be a numeric vector with a length equal to the number of features. Each element in the list corresponds to a feature and should take one of the following values:

  - 0: No monotonic constraint is imposed on the corresponding feature's partial response.
  - +1: The partial response of the corresponding feature should be monotonically increasing with respect to the target.
  - -1: The partial response of the corresponding feature should be monotonically decreasing with respect to the target.

`objective`
: The objective function to optimize. Current options include:

  - `"auto"` (try to determine automatically between `"log_loss"` and `"rmse"`).
  - `"rmse"` (root mean squared error).
  - `"poisson_deviance"` (e.g., for counts or non-negative integers).
  - `"tweedie_deviance:variance_power=1.5"` (e.g., for modeling total loss in insurance applications).
  - `"gamma_deviance"` (e.g., for positive continuous response).
  - `"pseudo_huber:delta=1.0"` (e.g., for robust regression).
  - `"rmse_log"` (`"rmse"` with a log link function).

  Default is `"auto"` which assumes `"log_loss"` if the response is a factor or character string and `"rmse"` otherwise. It's a good idea to always explicitly set this argument.

`n_jobs`
: Number of jobs to run in parallel. Default is -1. Negative integers are interpreted as following joblib's formula (`n_cpus + 1 + n_jobs`), just like scikit-learn. For example, `n_jobs = -2` means using all threads except 1.

`random_state`
: Random state. Setting to `NULL` generates non-repeatable sequences. Default is 42 to remain consistent with the corresponding Python module.

`...`
: Additional optional argument. (Currently ignored.)

Details

In short, EBMs have the general form

$$E\left[g\left(Y|\boldsymbol{x}\right)\right] = \theta_0 + \sum_i f_i\left(x_i\right) + \sum_{ij} f_{ij}\left(x_i, x_j\right) \quad \left(i \neq j\right),$$

where,

- g is a link function that allows the model to handle various response types (e.g., the logit link for logistic regression or Poisson deviance for modeling counts and rates);
- θ_0 is a constant intercept (or bias term); ?
- f_i is the term contribution (or shape function) for predictor x_i (i.e., it captures the main effect of x_i on $E[Y|\boldsymbol{x}]$);
- f_{ij} is the term contribution for the pair of predictors x_i and x_j (i.e., it captures the joint effect, or pairwise interaction effect of x_i and x_j on $E[Y|\boldsymbol{x}]$).

Value

An object of class "EBM" for which there are print, predict, plot, and merge methods.

Examples

```
## Not run:
  #
  # Regression example
  #

  # Fit a default EBM regressor
  fit <- ebm(mpg ~ ., data = mtcars, objective = "rmse")

  # Generate some predictions
  head(predict(fit, newdata = mtcars))
  head(predict(fit, newdata = mtcars, se_fit = TRUE))

  # Show global summary and GAM shape functions
  plot(fit)  # term importance scores
  plot(fit, term = "cyl")
  plot(fit, term = "cyl", interactive = TRUE)

  # Explain prediction for first observation
  plot(fit, local = TRUE, X = subset(mtcars, select = -mpg)[1L, ])

## End(Not run)
```

---

geom_stepribbon *Step ribbons and area plots*

Description

A combination of geom_ribbon() and geom_step().

Usage

```
geom_stepribbon(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

Source

Taken from ldatools.

`install_interpret` *Install interpret*

Description

This function will install interpret along with all of its dependencies.

Usage

```
install_interpret(
  envname = "r-ebm",
  ...,
  extra_packages = c("plotly>=3.8.1"),
  python_version = ">=3.9,<=3.12",
  restart_session = TRUE
)
```

Arguments

`envname`	Name of or path to a Python virtual environment.
`...`	Additional optional arguments. (Currently ignored.)
`extra_packages`	Additional Python packages to install alongside interpret.
`python_version`	Passed on to virtualenv_starter().
`restart_session`	Whether to restart the R session after installing (note this will only occur within RStudio).

Value

No return value, called for side effects.

`merge.EBM` *Merge method for EBM objects*

Description

Merge multiple EBMs together.

Usage

```
## S3 method for class 'EBM'
merge(x, y, ...)
```

Arguments

x, y	Fitted ebm objects that have been trained on similar data sets that have the same set of features.
...	Additional ebm objects to be merged.

Value

A merged ebm object.

Note

As of right now, the `merge()` function produces the following error message:

```
Error in py_repr(x) :
 AttributeError: 'ExplainableBoostingRegressor' object has no attribute 'cat_smooth'
Run `reticulate::py_last_error()` for details.
```

This seems to be a bug in the underlying interpret library and does not prevent this function from working. The error message is seemingly just a side effect.

Examples

```
## Not run:
# Generate list of EBMs with different random seeds
ebms <- lapply(1:3, FUN = function(i) {
  ebm(mpg ~ ., data = mtcars, outer_bags = 1, random_state = i, obj = "rmse")
})

# Merge EBMs into one and plot term contribution for `cyl`
merged <- do.call(merge, args = ebms)
plot(merged, term = "cyl")

## End(Not run)
```

---

plot.EBM *Interpret plots for fitted EBM objects*

Description

Provides an interactive visualization for a given explanation(s).

Usage

```
## S3 method for class 'EBM'
plot(
  x,
  term = NULL,
  local = FALSE,
  X = NULL,
  y = NULL,
  init_score = NULL,
  interactive = FALSE,
  n_terms = NULL,
  geom = c("point", "col", "bar"),
  mapping = NULL,
  aesthetics = list(),
  horizontal = FALSE,
  uncertainty = TRUE,
  width = 0.5,
  alpha = 0.5,
  fill = "grey",
  display = c("viewer", "markdown", "url"),
  viewer = c("browser", "rstudio"),
  full_dashboard = FALSE,
  ...
)
```

Arguments

x	A fitted ebm() object.
term	Character string specifying which term to plot. For interaction effect, you can supply a pair (e.g., term = c("x1", "x2")). Default is NULL which will just display the overall importance of each term.
local	Logocial indicating whether to display local explanations (TRUE) or global explanations (FALSE). Default is FALSE.
X	Data frame or matrix of samples. Unless display = "url" or full_dashboard = TRUE, then X can only contain a single row.
y	Optional vector of response values corresponding to X.
init_score	Optional. Either a model that can generate scores or per-sample initialization score. If samples scores it should be the same length as X.
interactive	Logical indicating whether to produce an interactive plot based on HTML. Default is FALSE. Currently, only interactive graphics (i.e., interactive = TRUE) are available for multiclass outcomes.
n_terms	Integer specifying the maximum number of variable importance scores to plot. Default is NULL which corresponds to all terms in the fitted model.
geom	Character string specifying which type of plot to construct for terms associated with categorical features. Current options are:

- `geom = "bar"` (or `"col"`) uses geom_col to construct a bar chart of the scores.
- `geom = "point"` uses geom_point to construct a Cleveland dot plot of the term scores.

Default is `"point"`.

`mapping` Set of aesthetic mappings created by aes-related functions and/or tidy eval helpers. See example usage below.

`aesthetics` List specifying additional arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like`colour = "red"` or `size = 3`. See example usage below.

`horizontal` Logical indicating whether or not term plots for categorical features should be flipped horzintally. Default is `FALSE`.

`uncertainty` Logical indicating whether or not to also display uncertainty via error bars on the main effect plots. Default is `TRUE`. Not very useful unless `outer_bags > 1` when calling `ebm()`.

`width` Numeric specifying the width of the error bars displayed in bar/ dot plots for categorical features. Default is 0.5.

`alpha` Numeric between 0 and 1 specifying the level of transparency to use when displaying uncertainty in plots for continuous features. Default is 0.5.

`fill` Character string specifying the fill color to use when displaying uncertainty in plots for continuous features. Default is `"grey"`.

`display` Character string specifying how the results should be displayed whenever `interactive = TRUE`. Available options are `"viewer"` (e.g., RStudio viewer browser), `"markdown"` (e.g., for vingettes, Quarto, or Rmarkdown documents), or `"url"` (e.g., to print a URL which can be pasted into a browser). When `display = "url"`, a URL for viewing the entire interpret dashboard is provided (i.e., the `term` and `full_dashboard` arguments are ignored).

`viewer` Character string specifying how the results should be viewed. Current choices are `"broswer"`, which calls `utils::browseURL()` to display the results in an HTML browser, or `"rstudio"` for displaying the results within the Viewer pane in an active RStudio session. Also works in VS Code. Default is `"browser"`.

`full_dashboard` Logical indicating whether or not to display the full interpret dashboard. Default is `FALSE`. Only works when `display = "viewer"` or `display = "url"` (e.g., paste the resulting URL in your browser).

`...` Additional optional arguments. Currently only passed onto levelplot() for heatmaps of interaction effects.

Value

When `interactive = FALSE` (the default), the output is either a ggplot object when visualizing term importance scores or main effects, or a trellis object when visualizing pairwise interaction effects. When `interactive = TRUE`, the return value depends on `display` argument. When `display = "url"`, a character string is returned giving the URL for displaying the HTML-based visualization. Otherwise, the results are viewed as requested (i.e., in a browser, built-in viewer, or displayed in rendered HTML output).

predict.EBM *Predict method for EBM objects*

Description

Compute predicted values from a fitted explainable boosting machine.

Usage

```
## S3 method for class 'EBM'
predict(
  object,
  newdata,
  type = c("response", "link", "class", "terms"),
  se_fit = FALSE,
  init_score = NULL,
  ...
)
```

Arguments

object	A fitted ebm object.
newdata	A data frame in which to look for variables with which to predict.
type	The type of prediction required. Current options include: • "response": Returns predictions on the scale of the response variable. Thus, for a categorical outcome (i.e., binary or multiclass), a matrix of predicted probabilities is returned. • "link": Returns predictions on the link scale. For a binary outcome with logit link, for example, this results in a vector of logits. For a multiclass outcome, this will return a matrix with one column for each class. Ignored for regression problems. • "class": Returns a vector predicted class label for categorical outcomes. • "terms": Returns a matrix (or list of matrices for multiclass outcomes) of the individual term contributions (e.g., the f(x)'s). Note that term contributions are on the link scale, where they are additive.
se_fit	Logical indicating whether or not standard errors are required. Ignored for multiclass outcomes. Note that standard errors are only available on the link scale.
init_score	Optional. Either a model that can generate scores or per-sample initialization score. If samples scores it should be the same length as newdata.
...	Additional optional arguments. (Currently ignored.)

Value

Either a vector, matrix, or list of results. See the type argument for details.

`print.EBM` *Print model summary*

Description

Display basic information about a fitted ebm object.

Usage

```
## S3 method for class 'EBM'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	A fitted ebm object.
`digits`	The number of significant digits to be passed to `format()` when printing.
`...`	Additional optional arguments to be passed to `print.default()`.

Value

Invisibly returns the printed ebm object.

Index