Package: glex (via r-universe)

October 18, 2024

Type Package

Title Global Explanations for Tree-Based Models

Version 0.4.0.9000

Description Global explanations for tree-based models by decomposing regression or classification functions into the sum of main components and interaction components of arbitrary order. Calculates SHAP values and q-interaction SHAP for all values of q for tree-based models such as xgboost.

License GPL-3

URL `https://github.com/PlantedML/glex`, `http://plantedml.com/glex/`

BugReports `https://github.com/PlantedML/glex/issues`

Imports checkmate, data.table, foreach, ggplot2, Rcpp (>= 1.0.8), scico, stats, utils

Suggests covr, doParallel, ISLR2, knitr, patchwork, randomPlantedForest, rmarkdown, testthat (>= 3.0.0), xgboost, ranger

LinkingTo Rcpp

VignetteBuilder knitr

Remotes PlantedML/randomPlantedForest

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 2.10)

LazyData true

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

RemoteUrl https://github.com/PlantedML/glex

RemoteRef HEAD

RemoteSha 7cdc418ef726311cccf852559efaef406f7896f2

Contents

`autoplot.glex` *Plot Prediction Components*

Description

Plotting the main effects among the prediction components is effectively identical to a partial dependence plot, centered to 0.

Usage

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

plot_main_effect(object, predictor, rug_sides = "b", ...)

plot_threeway_effects(object, predictors, rug_sides = "b", ...)

plot_twoway_effects(object, predictors, rug_sides = "b", ...)
```

Arguments

`object` Object of class `glex`.

`...` Used for future expansion.

`predictor, predictors`
(`character`) vector of predictor names, e.g. `"x1"` to plot main effect of `x1`, and `c("x1", "x2")` to plot the interaction term `x1:x2`.

`rug_sides` (`character(1): "b"`) Sides to plot rug (see `ggplot2::geom_rug()`) plot on for continuous predictors.. Default is `"b"` for both sides. Set to `"none"` to disable rug plot.

Value

A `ggplot2` object.

See Also

plot_pdp()

Other Visualization functions: autoplot.glex_vi(), glex_explain(), plot_pdp()

Examples

```
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)

# introduce factor variables to show categorical feature handling
mtcars$cyl <- factor(mtcars$cyl)
mtcars$vs <- factor(mtcars$vs)

# Fit forest, get components
set.seed(12)
rpfit <- rpf(mpg ~ cyl + wt + hp + drat + vs, data = mtcars, ntrees = 25, max_interaction = 3)
components <- glex(rpfit, mtcars)

# Main effects ----
plot_main_effect(components, "wt")
plot_main_effect(components, "cyl")
}
# plot_threeway_effects(components, c("hr", "temp", "workingday"))
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)

# 2-degree interaction effects ----
# 2d continuous, scatterplot of arbitrary orientation
plot_twoway_effects(components, c("wt", "drat"))
# flipped: plot_twoway_effects(components, c("drat", "wt"))

# continuous + categorical (forces continuous on x axis, colors by categorical)
plot_twoway_effects(components, c("wt", "cyl"))
# identical: plot_twoway_effects(components, c("cyl", "wt"))

# 2d categorical, heatmap of arbitrary orientation
plot_twoway_effects(components, c("vs", "cyl"))
plot_twoway_effects(components, c("cyl", "vs"))
}
```

---

autoplot.glex_vi *Plot glex Variable Importances*

Description

Plot glex Variable Importances

Usage

```
## S3 method for class 'glex_vi'
autoplot(
  object,
  by_degree = FALSE,
  threshold = 0,
  max_interaction = NULL,
  scale = "absolute",
  ...
)
```

Arguments

`object` Object of class `glex_vi`, see `glex_vi()`.

`by_degree` (`logical(1)`: `FALSE`) Optionally sum values by degree of interaction, resulting in one contribution score for all main effects, all second-order interactions, etc.

`threshold` (`numeric(1)`: 0) Optional threshold to filter output to include only importance scores greater than this value. Refers to the chosen `scale`.

`max_interaction` (`integer(1)`: `NULL`) Optionally filter plot to show terms up to the specified degree of interaction. Similar to `threshold`, all other terms will be aggregated under a `"Remaining terms"` label.

`scale` (`"absolute"`) Plot average absolute contributions (default) or the same value but scaled by the average prediction (`"relative"`).

`...` (Unused)

Value

A `ggplot` object.

See Also

glex_vi

Other Visualization functions: `autoplot.glex`(), `glex_explain`(), `plot_pdp`()

bike *Bikesharing data*

Description

A reduced version of the `Bikeshare` data as included with `ISLR2`. The dataset has been converted to a `data.table`, with the following changes:

Usage

```
bike
```

Format

An object of class `data.table` (inherits from `data.frame`) with 8645 rows and 11 columns.

Details

- `hr` has been copnverted to a numeric
- `workingday` was recoded to a binary `factor` with labels `c("No Workingday", "Workingday")`
- `season` was recoded to a `factor` with labels `c("Winter", "Spring", "Summer", "Fall")`
- Variables `atemp`, `day`, `registered` and `casual` were removed

Source

`Bikeshare` in package `ISLR2`

`glex` *Global explanations for tree-based models.*

Description

Global explanations for tree-based models by decomposing regression or classification functions into the sum of main components and interaction components of arbitrary order. Calculates SHAP values and q-interaction SHAP for all values of q for tree-based models such as xgboost.

Usage

```
glex(
  object,
  x,
  max_interaction = NULL,
  features = NULL,
  probFunction = NULL,
  ...
)

## S3 method for class 'rpf'
glex(object, x, max_interaction = NULL, features = NULL, ...)

## S3 method for class 'xgb.Booster'
glex(
  object,
  x,
  max_interaction = NULL,
  features = NULL,
  probFunction = NULL,
  ...
)
```

```
## S3 method for class 'ranger'
glex(
  object,
  x,
  max_interaction = NULL,
  features = NULL,
  probFunction = NULL,
  ...
)
```

Arguments

`object` Model to be explained, either of class `xgb.Booster` or `rpf`.

`x` Data to be explained.

`max_interaction`
`(integer(1): NULL)`
Maximum interaction size to consider. Defaults to using all possible interactions available in the model.
For `xgboost`, this defaults to the `max_depth` parameter of the model fit.
If not set in `xgboost`, the default value of `6` is assumed.

`features` Vector of column names in `x` to calculate components for. Default is `NULL`, i.e. all features are used.

`probFunction` Either "path-dependent" to use old path-dependent weighting of leaves or a user specified probability function of the signature function(coords, lb, ub). Defaults to `NULL` or "emprical", i.e. the empirical marginal probabilities will be used

`...` Further arguments passed to methods.

Details

For parallel execution using `xgboost` models, register a backend, e.g. with `doParallel::registerDoParallel()`.

Value

Decomposition of the regression or classification function. A `list` with elements:

- `shap`: SHAP values (`xgboost` method only).
- `m`: Functional decomposition into all main and interaction components in the model, up to the degree specified by `max_interaction`. The variable names correspond to the original variable names, with `:` separating interaction terms as one would specify in a `formula` interface.
- `intercept`: Intercept term, the expected value of the prediction.

Examples

```
# Random Planted Forest -----
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)
rp <- rpf(mpg ~ ., data = mtcars[1:26, ], max_interaction = 2)

glex_rpf <- glex(rp, mtcars[27:32, ])
str(glex_rpf, list.len = 5)
}
# xgboost -----
if (requireNamespace("xgboost", quietly = TRUE)) {
library(xgboost)
x <- as.matrix(mtcars[, -1])
y <- mtcars$mpg
xg <- xgboost(data = x[1:26, ], label = y[1:26],
              params = list(max_depth = 4, eta = .1),
              nrounds = 10, verbose = 0)
glex(xg, x[27:32, ])

## Not run:
# Parallel execution
doParallel::registerDoParallel()
glex(xg, x[27:32, ])

## End(Not run)
}
# ranger -----
if (requireNamespace("ranger", quietly = TRUE)) {
library(ranger)
x <- as.matrix(mtcars[, -1])
y <- mtcars$mpg
rf <- ranger(x = x[1:26, ], y = y[1:26],
             num.trees = 5, max.depth = 3,
             node.stats = TRUE)
glex(rf, x[27:32, ])

## Not run:
# Parallel execution
doParallel::registerDoParallel()
glex(rf, x[27:32, ])

## End(Not run)
}
```

glex_explain *Explain a single prediction*

Description

Plots the prediction components for a single observation, identified by the row number in the dataset used with `glex()`. Since the resulting plot can be quite busy due to potentially large amounts of elements, it is highly recommended to use `predictors`, `max_interaction`, or `threshold` to restrict the number of elements in the plot.

Usage

```
glex_explain(
  object,
  id,
  threshold = 0,
  max_interaction = NULL,
  predictors = NULL,
  class = NULL,
  barheight = 0.5
)
```

Arguments

`object` Object of class `glex` containing prediction components and data to be explained.

`id` (`integer(1)`) Row ID of the observation to be explained in `object$x`.

`threshold` (`numeric(1): 0`) Threshold to filter output by in case of many negligible effects.

`max_interaction` (`integer(1): NULL`) Optionally filter plot to show terms up to the specified degree of interaction. Similar to `threshold`, all other terms will be aggregated under a `"Remaining terms"` label.

`predictors` (`character: NULL`) Vector of column names in `$x` to restrict plot to.

`class` (`character: NULL`) For multiclass targets, specifies the target class to limit output.

`barheight` (`numeric(1): 0.5`) Relative height of horizontal bars. Preferred value may depend on the number of vertical elements, hence it may be necessary to adjust this value as needed.

Value

A ggplot object.

See Also

Other Visualization functions: `autoplot.glex()`, `autoplot.glex_vi()`, `plot_pdp()`

Examples

```
set.seed(1)
# Random Planted Forest -----
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)

rp <- rpf(mpg ~ ., data = mtcars[1:26, ], max_interaction = 2)

glex_rpf <- glex(rp, mtcars[27:32, ])

glex_explain(glex_rpf, id = 3, predictors = "hp", threshold = 0.01)
}
```

glex_vi — Variable Importance for Main and Interaction Terms

Description

Variable Importance for Main and Interaction Terms

Usage

```
glex_vi(object, ...)
```

Arguments

`object`	Object of class `glex`.
`...`	(Unused)

Details

The `m` reported here is the average absolute value of `m` as reported by `glex()`, aggregated by `term`:

$$\texttt{m} = \frac{1}{n}\sum_{i=1}^{n}|m|$$

In turn, `m_rel` rescales `m` by the average prediction of the model (m_0, `intercept` as reported by `glex()`):

$$\texttt{m\_rel} = \frac{\texttt{m}}{m_0}$$

Value

A `data.table` with columns:

- `degree` (`integer`): Degree of interaction of the `term`, with `1` being main effects, `2` being 2-degree interactions etc.
- `term` (`character`): Model term, e.g. main effect `x1` or interaction term `x1:x2`, `x1:x3:x5` etc.
- `class` (`factor`): For multiclass targets only: The associated target class. Lists all classes in the target, not limited to the majority vote.
- `m` (`numeric`): Average absolute contribution of `term`, see Details.
- `m_rel` (`numeric`): `m` but relative to the average prediction (`intercept` in `glex()` output).

See Also

autoplot.glex_vi

Examples

```
set.seed(1)
# Random Planted Forest -----
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)

rp <- rpf(mpg ~ ., data = mtcars[1:26, ], max_interaction = 3)

glex_rpf <- glex(rp, mtcars[27:32, ])

# All terms
vi_rpf <- glex_vi(glex_rpf)

library(ggplot2)
# Filter to contributions greater 0.05 on the scale of the target
autoplot(vi_rpf, threshold = 0.05)
# Summarize by degree of interaction
autoplot(vi_rpf, by_degree = TRUE)
# Filter by relative contributions greater 0.1%
autoplot(vi_rpf, scale = "relative", threshold = 0.001)
}

# xgboost -----
if (requireNamespace("xgboost", quietly = TRUE)) {
library(xgboost)
x <- as.matrix(mtcars[, -1])
y <- mtcars$mpg
xg <- xgboost(data = x[1:26, ], label = y[1:26],
              params = list(max_depth = 4, eta = .1),
              nrounds = 10, verbose = 0)
glex_xgb <- glex(xg, x[27:32, ])
vi_xgb <- glex_vi(glex_xgb)

library(ggplot2)
autoplot(vi_xgb)
autoplot(vi_xgb, by_degree = TRUE)
}
```

plot_pdp *Partial Dependence Plot*

Description

A version of plot_main_effect with the intercept term (horizontal line) added, resulting in a partial dependence plot.

Usage

```
plot_pdp(object, predictor, rug_sides = "b", ...)
```

Arguments

`object`	Object of class `glex`.
`predictor`	(`character(1)`) predictor names, e.g. `"x1"` to plot main effect of `x1`.
`rug_sides`	(`character(1): "b"`) Sides to plot rug (see `ggplot2::geom_rug()`) plot on for continuous predictors.. Default is `"b"` for both sides. Set to `"none"` to disable rug plot.
`...`	Used for future expansion.

Value

A `ggplot2` object.

See Also

`plot_main_effect()`

Other Visualization functions: `autoplot.glex()`, `autoplot.glex_vi()`, `glex_explain()`

Examples

```
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)

# introduce factor variables to show categorical feature handling
mtcars$cyl <- factor(mtcars$cyl)
mtcars$vs <- factor(mtcars$vs)

# Fit forest, get components
set.seed(12)
rpfit <- rpf(mpg ~ cyl + wt + hp + drat + vs, data = mtcars, ntrees = 25, max_interaction = 3)
components <- glex(rpfit, mtcars)

plot_pdp(components, "wt")
plot_pdp(components, "cyl")
}
```

`print.glex` *Print glex objects*

Description

This is implemented mainly to avoid flooding the console in cases where the `glex` object uses many terms, which leads to a large amount of column names of `$m` being printed to the console. This function wraps `str()` with a truncated output for a more compact representation.

Usage

```
## S3 method for class 'glex'
print(x, ...)
```

Arguments

x	Object to print.
...	(Unused)

Examples

```
# Random Planted Forest -----
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)
rp <- rpf(mpg ~ hp + wt + drat, data = mtcars[1:26, ], max_interaction = 2)

glex(rp, mtcars[27:32, ])
}
```

subset_components *Subset components*

Description

Subset components

Usage

```
subset_components(components, term)

subset_component_names(components, term)
```

Arguments

components	An object of class glex.
term	(character(1)) A main term name to subset by, e.g. "x1".

Value

- subset_components: An object of class glex.
- subset_component_names: A character vector.

Examples

```
if (requireNamespace("randomPlantedForest", quietly = TRUE)) {
library(randomPlantedForest)

# introduce factor variables to show categorical feature handling
mtcars$cyl <- factor(mtcars$cyl)
mtcars$vs <- factor(mtcars$vs)

# Fit forest, get components
set.seed(12)
rpfit <- rpf(mpg ~ cyl + wt + hp + drat + vs, data = mtcars, ntrees = 25, max_interaction = 3)
components <- glex(rpfit, mtcars)

# Get component object with only "hp" and its interactions
subset_components(components, "hp")

subset_component_names(components, "hp")
}
```

theme_glex *A ggplot2 theme for glex plots*

Description

This is a slight variation of `ggplot2::theme_minimal()` with increased font size.

Usage

```
theme_glex(
  base_size = 13,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  grid_x = TRUE,
  grid_y = FALSE
)
```

Arguments

`base_size` (13) Base font size, given in pts.

`base_family` ("") Base font family

`base_line_size`, `base_rect_size`
(`base_size / 22`) Base size for line and rect elements

`grid_x` (`TRUE`) Display horizontal grid lines?

`grid_y` (`FALSE`) Display vertical grid lines?

Value

A `ggplot2` theme object

Examples

```
library(ggplot2)

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_glex()
```

Index