Package ‘breakDown’

March 12, 2024

**Title**  Model Agnostic Explainers for Individual Predictions

**Version**  0.2.2

**Description**  Model agnostic tool for decomposition of predictions from black boxes. Break Down Table shows contributions of every variable to a final prediction. Break Down Plot presents variable contributions in a concise graphical way. This package work for binary classifiers and general regression models.

**Depends**  R (>= 3.0)

**Date**  2024-03-11

**License**  GPL-2

**Encoding**  UTF-8

**LazyData**  true

**Imports**  ggplot2

**Suggests**  knitr, rmarkdown, e1071, kernlab, xgboost, caret, randomForest, DALEX, ranger, testthat

**VignetteBuilder**  knitr

**URL**  https://pbiecek.github.io/breakDown/

**BugReports**  https://github.com/pbiecek/breakDown/issues

**NeedsCompilation**  no

**Author**  Przemyslaw Biecek [aut, cre], Aleksandra Grudziaz [ctb]

**Maintainer**  Przemyslaw Biecek <przemyslaw.biecek@gmail.com>

**Repository**  CRAN

**Date/Publication**  2024-03-11 23:30:02 UTC

**R topics documented:**

```
betas ................................................................. 2
break_down .......................................................... 2
broken .............................................................. 4
```
betas

Extract betas values of a model for specific observations

Usage

betas(object, newdata, ...)

Arguments

object a model
newdata new observation(s) with columns that correspond to variables used in the model
... unused additional parameters

Author(s)

Joseph Larmarange

break_down

Model Agnostic Experimental Approach to Break Down Plots with Interactions

Description

This function implements decomposition of model predictions with identification of interactions. The complexity of this function is \( O(2^p) \) for additive models and \( O(2^p p^2) \) for interactions. This function works in similar way to step-up and step-down greedy approximations, the main difference is that in the fisrt step the order of variables is determied. And in the second step the impact is calculated.
Usage

break_down(
  explainer,
  new_observation,
  check_interactions = TRUE,
  keep_distributions = FALSE
)

Arguments

explainer a model to be explained, preprocessed by function ‘DALEX::explain()’. 
new_observation a new observation with columns that corresponds to variables used in the model 
check_interactions 
the orgin/baseline for the “breakDown” plots, where the rectangles start. It may be a number or a character "Intercept". In the latter case the orgin will be set to model intercept. 
keep_distributions 
if TRUE, then the distribution of partial predictions is stored in addition to the average. 

Value

an object of the broken class 

Examples

## Not run: 
library("DALEX") 
library("breakDown") 
library("randomForest") 
set.seed(1313) 
# example with interaction 
# classification for HR data 
model <- randomForest(status ~ . , data = HR) 
new_observation <- HRTest[1,] 
data <- HR[1:1000,] 
predict.function <- function(m,x) predict(m,x, type = "prob")[,1] 
explainer_rf_fired <- explain(model, 
  data = HR[1:1000,1:5], 
y = HR$status[1:1000] == "fired", 
predict_function = function(m,x) predict(m,x, type = "prob")[,1], 
label = "fired") 

bd_rf <- break_down(explainer_rf_fired, 
  new_observation, 
  keep_distributions = TRUE) 

bd_rf
plot(bd_rf)
plot(bd_rf, plot_distributions = TRUE)

bd_rf <- break_down(explainer_rf_fired,
    new_observation,
    check_interactions = FALSE,
    keep_distributions = TRUE)

bd_rf
plot(bd_rf)

# example for regression - apartment prices
# here we do not have interactions
model <- randomForest(m2.price ~ ., data = apartments)
explainer_rf <- explain(model,
    data = apartmentsTest[1:1000,2:6],
    y = apartmentsTest$m2.price[1:1000],
    label = "rf")

bd_rf <- break_down(explainer_rf,
    apartmentsTest[1,],
    check_interactions = FALSE,
    keep_distributions = TRUE)

bd_rf
plot(bd_rf)
plot(bd_rf, plot_distributions = TRUE)

## End(Not run)

---

**broken**

*Generic Function for Breaking Down of Model Predictions*

**Description**

The broken function is a generic function for decomposition of model predictions. For linear models please use `broken.lm`, for generic linear models please use `broken.glm`. For all other models please use the model agnostic version `broken.default`. Please note, that some of these functions have additional parameters.

**Usage**

`broken(model, new_observation, ...)`

**Arguments**

- `model` a model
- `new_observation` a new observation with columns that corresponds to variables used in the model
- `...` other parameters
Value

an object of the broken class

Examples

## Not run:
library("breakDown")
library("randomForest")
library("ggplot2")
set.seed(1313)
model <- randomForest(factor(left)~., data = HR_data, family = "binomial", maxnodes = 5)
predict.function <- function(model, new_observation)
  predict(model, new_observation, type="prob")[,2]
predict.function(model, HR_data[11,-7])
explain_1 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "down")
explain_1
plot(explain_1) + ggtitle("breakDown plot (direction=down) for randomForest model")

explan_2 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "down", keep_distributions = TRUE)
plot(explain_2, plot_distributions = TRUE) +
  ggtitle("breakDown distributions (direction=down) for randomForest model")

explain_3 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "up", keep_distributions = TRUE)
plot(explain_3, plot_distributions = TRUE) +
  ggtitle("breakDown distributions (direction=up) for randomForest model")

## End(Not run)

broken.default

Model Agnostic Approach to Breaking Down of Model Predictions

Description

This function implements two greedy strategies for decompositions of model predictions (see the
direction parameter). Both strategies are model agnostic, they are greedy but in most cases they
give very similar results. Find more information about these strategies in https://arxiv.org/
abs/1804.01955.

Usage

## Default S3 method:
broken(
  model,
  new_observation,
  data,
  direction = "up",
)
arguments

model a model, it can be any predictive model, find examples for most popular frameworks in vignettes
new_observation a new observation with columns that corresponds to variables used in the model
data the original data used for model fitting, should have same columns as the ‘new_observation’.
direction either ‘up’ or ‘down’ determined the exploration strategy
... other parameters
baseline the origin/baseline for the breakDown plots, where the rectangles start. It may be a number or a character "Intercept". In the latter case the origin will be set to model intercept.
keep_distributions if TRUE, then the distribution of partial predictions is stored in addition to the average.
predict.function function that will calculate predictions out of model. It shall return a single numeric value per observation. For classification it may be a probability of the default class.

value

an object of the broken class

examples

## Not run:
library("breakDown")
library("randomForest")
library("ggplot2")
set.seed(1313)
model <- randomForest(factor(left) ~ ., data = HR_data, family = "binomial", maxnodes = 5)
predict.function <- function(model, new_observation)
  predict(model, new_observation, type="prob")[,2]
predict.function(model, HR_data[11,-7])
explain_1 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "down")
explain_1
plot(explain_1) + ggtitle("breakDown plot (direction=down) for randomForest model")

explain_2 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "down", keep_distributions = TRUE)
plot(explain_2, plot_distributions = TRUE) +
explain_3 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "up", keep_distributions = TRUE)
plot(explain_3, plot_distributions = TRUE) +
  ggtitle("breakDown distributions (direction=up) for randomForest model")

## End(Not run)

broken.glm

---

**Description**

Breaking Down of Model Predictions for glm models

**Usage**

```r
## S3 method for class 'glm'
broken(
  model,
  new_observation,
  ..., 
  baseline = 0,
  predict.function = stats::predict.glm
)
```

**Arguments**

- `model`: a glm model
- `new_observation`: a new observation with columns that corresponds to variables used in the model
- `...`: other parameters
- `baseline`: the origin/baseline for the breakDown plots, where the rectangles start. It may be a number or a character "Intercept". In the latter case the origin will be set to model intercept.
- `predict.function`: function that will calculate predictions out of model (typically `predict` or `betas`)

**Value**

an object of the broken class
Examples

```r
# example for wine data
wine$qualityb <- factor(wine$quality > 5.5, labels = c("bad", "good"))
modelg <- glm(qualityb ~ fixed.acidity + volatile.acidity + citric.acid +
              residual.sugar + chlorides + free.sulfur.dioxide +
              total.sulfur.dioxide + density + pH + sulphates + alcohol,
              data = wine, family = "binomial")
new_observation <- wine[1,]
br <- broken(modelg, new_observation)
logit <- function(x) exp(x)/(1+exp(x))
plot(br, logit)

# example for HR_data
model <- glm(left ~ ., data = HR_data, family = "binomial")
explain_1 <- broken(model, HR_data[1,])
plot(explain_1)
plot(explain_1, trans = function(x) exp(x)/(1+exp(x)))

explain_2 <- broken(model, HR_data[1,], predict.function = betas)
plot(explain_2, trans = function(x) exp(x)/(1+exp(x)))
```

---

### broken.lm

**Description**

Breaking Down of Model Predictions for `lm` models

**Usage**

```r
## S3 method for class 'lm'
broken(
  model,
  new_observation,
  ..., baseline = 0,
  predict.function = stats::predict.lm
)
```

**Arguments**

- `model` a `lm` model
- `new_observation` a new observation with columns that correspond to variables used in the model
- `...` other parameters
baseline
the origin/baseline for the breakDown plots, where the rectangles start. It may be a number or a character "Intercept". In the latter case the origin will be set to model intercept.

predict.function
function that will calculate predictions out of model (typically predict or betas)

Value
an object of the broken class

Examples
model <- lm(Sepal.Length~., data=iris)
new_observation <- iris[1,]
br <- broken(model, new_observation)
plot(br)

# works for interactions as well
model <- lm(Sepal.Length ~ Petal.Width*Species, data = iris)
summary(model)

new_observation <- iris[1,]
br <- broken(model, new_observation)
br
plot(br)

br2 <- broken(model, new_observation, predict.function = betas)
br2
plot(br2)

HR_data

Why are our best and most experienced employees leaving prematurely?

Description
A dataset from Kaggle competition Human Resources Analytics. https://www.kaggle.com/

Format
A data frame with 14999 rows and 10 variables

Details
• satisfaction_level Level of satisfaction (0-1)
• last_evaluation Time since last performance evaluation (in Years)
• number_project Number of projects completed while at work
• average_monthly_hours Average monthly hours at workplace
• time_spend_company Number of years spent in the company
• Work_accident Whether the employee had a workplace accident
• left Whether the employee left the workplace or not (1 or 0) Factor
• promotion_last_5years Whether the employee was promoted in the last five years
• sales Department in which they work for
• salary Relative level of salary (high)

Source

Dataset HR-analytics from https://www.kaggle.com

---

plot.broken       Break Down Plot

Description

Break Down Plot

Usage

```r
## S3 method for class 'broken'
plot(
  x,
  trans = I,
  ...,
  top_features = 0,
  min_delta = 0,
  add_contributions = TRUE,
  vcolors = c("-1" = "#f05a71", "0" = "#371ea3", "1" = "#8bdcbe", X = "#371ea3"),
  digits = 3,
  rounding_function = round,
  plot_distributions = FALSE
)
```

Arguments

- `x` the model model of 'broken' class
- `trans` transformation that shall be applied to scores
- `...` other parameters
- `top_features` maximal number of variables from model we want to plot
- `min_delta` minimal stroke value of variables from model we want to plot
- `add_contributions` shall variable contributions to be added on plot?
- `vcolors` named vector with colors
digits

number of decimal places (round) or significant digits (signif) to be used. See the rounding_function argument

rounding_function

function that is to used for rounding numbers. It may be \texttt{signif()} which keeps a specified number of significant digits. Or the default \texttt{round()} to have the same precision for all components

plot_distributions

if \texttt{TRUE} then distributions of conditional propotions will be plotted. This requires keep_distributions=\texttt{TRUE} in the \texttt{broken.default()}.

Value

a \texttt{ggplot2} object

Examples

```r
## Not run:
library("breakDown")
library("randomForest")
library("ggplot2")
set.seed(1313)
model <- randomForest(factor(left)~., data = HR_data, family = "binomial", maxnodes = 5)
predict.function <- function(model, new_observation)
  predict(model, new_observation, type="prob")[,2]
predict.function(model, HR_data[11,-7])
extain_1 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "down")
extain_1
plot(explain_1) + ggtitle("breakDown plot (direction=down) for randomForest model")

explain_2 <- broken(model, HR_data[11,-7], data = HR_data[,-7],
predict.function = predict.function, direction = "up", keep_distributions = TRUE)
plot(explain_3, plot_distributions = TRUE) +
ggtitle("breakDown distributions (direction=up) for randomForest model")

model <- lm(quality~., data=wine)
new_observation <- wine[1,]
br <- broken(model, new_observation)
plot(br)
plot(br, top_features = 2)
plot(br, top_features = 2, min_delta = 0.01)

## End(Not run)
```
Break Down Print

Description

Break Down Print

Usage

```r
## S3 method for class 'broken'
print(x, ..., digits = 3, rounding_function = round)
```

Arguments

- `x`: the model model of 'broken' class
- `...`: other parameters
- `digits`: number of decimal places (round) or significant digits (signif) to be used. See the rounding_function argument
- `rounding_function`: function that is to used for rounding numbers. It may be `signif()` which keeps a specified number of significant digits. Or the default `round()` to have the same precision for all components

Value

a data frame

---

Wine

White Wine Quality Data

Description

White wine quality data related to variants of the Portuguese "Vinho Verde" wine. For more details, consult: http://www.vinhoverde.pt/en/ or the reference Cortez et al., 2009.

Format

A data frame with 4898 rows and 12 variables
Details

A dataset downloaded from UCI Machine Learning Database archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv

- fixed.acidity
- volatile.acidity
- citric.acid
- residual.sugar
- chlorides
- free.sulfur.dioxide
- total.sulfur.dioxide
- density
- pH
- sulphates
- alcohol
- quality

Source

Index

∗ datasets
   HR_data, 9
   betas, 2
   break_down, 2
   broken, 4
   broken.default, 4, 5
   broken.glm, 4, 7
   broken.lm, 4, 8
   HR_data, 9
   plot.broken, 10
   print.broken, 12

wine, 12