Package 'uplifteval'

June 15, 2019

Type Package

Title Uplift Model Evaluation with Plots and Metrics

Version 0.1.0

Author Roland Stevenson

Maintainer Roland Stevenson <roland@rmg-services.com>

Description Provides a variety of plots and metrics to evaluate
uplift models including the 'R uplift' package's Qini metric and Qini plot,
a port of the 'python pylift' module's plotting function, and
an alternative plot (in beta) useful for continuous outcomes.
Background: Radcliffe (2007) <https://pdfs.semanticscholar.org/147b/32f3d56566c8654a9999c5477dded233328e.pdf>.

License GPL-3

URL https://github.com/ras44/uplifteval

BugReports https://github.com/ras44/uplifteval/issues

Encoding UTF-8

LazyData true

Suggests testthat, knitr, rmarkdown, grf, tweedie, qpdf

Depends R (>= 3.0.0),

Imports ggplot2, whisker, gridExtra, dplyr

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-06-15 07:50:03 UTC

R topics documented:

`new_PlUpliftEval` *A non-S3 "constructor" function that returns a list representing a pylift uplift eval object, PlUpliftEval. This object contains metrics and can be used to generate plots.*

Description

A non-S3 "constructor" function that returns a list representing a pylift uplift eval object, PlUpliftEval. This object contains metrics and can be used to generate plots.

Usage

```
new_PlUpliftEval(treatment = integer(), outcome = integer(),
  prediction = numeric(), p = "infer", n_bins = 20)
```

Arguments

`treatment`	numeric vector of treatment identifiers
`outcome`	numeric vector of outcomes
`prediction`	numeric vector of uplift predictions
`p`	optional "infer", numeric, numeric vector representing treatment propensities
`n_bins`	integer number of bins on x-axis; default 20

Value

a list representing a pylift uplift eval object

`plot_uplift` *Creates an uplift plot of cumulative differential treatment/control outcomes versus model score. Also provides a selection of metrics: max uplift as pct of total control outcome, optimum users targeted and optimum score targeting range.*

Description

Creates an uplift plot of cumulative differential treatment/control outcomes versus model score. Also provides a selection of metrics: max uplift as pct of total control outcome, optimum users targeted and optimum score targeting range.

Usage

```
plot_uplift(p1, W, Y, ns = min(table(W)), n_bs = 1, W_label = W,
  p0 = rep(0, length(p1)), balanced = TRUE, replace = TRUE,
  x_interval = 0.1, ...)
```

Arguments

p1	numeric vector of uplift predictions; can also be predicted outcomes for treated case (in this case p0 should contain predicted outcomes for the control case)
W	binary vector 1,0 of treatment assignments
Y	numeric vector of responses
ns	integer number of samples per bootstrap iteration; default min(table(W))
n_bs	integer number of bootstrap iterations
W_label	optional labels for the treatment options (default W)
p0	optional numeric vector of predicted outcomes for control case
balanced	optional boolean whether to sample equal proportions from treatment and control cases; default TRUE
replace	optional boolean whether to use replacement when sampling; default TRUE
x_interval	optional numeric the interval with which to split the
...	additional arguments (unused) x-axis

Examples

```
set.seed(0)
rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p <- 3
beta <- -0.5
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.5)
Y <- rl(pmax(beta+X[,1], 0) * W + X[,2])
p1 <- 1/(1+exp(-(beta+X[,1])))
plot_uplift(p1, W, Y, n_bs=20, x_interval = 0.05, balanced = TRUE)
```

```
set.seed(0)
n <- 2000; p <- 3
beta <- -0.5
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.5)
Y <- pmax(beta+X[,1], 0) * W + X[,2]
p1 <- 1/(1+exp(-(beta+X[,1])))
plot_uplift(p1, W, Y, n_bs=20, x_interval = 0.05, balanced = TRUE)
```

```
library(grf)
set.seed(123)

rl <- function(x){
  round(1/(1+exp(-x)))
}
n = 2000; p = 10
X = matrix(rnorm(n*p), n, p)
W = rbinom(n, 1, 0.2)
Y = rl(rl(X[,1]) * W - rl(X[,3]) * W + rnorm(n))
tau.forest = causal_forest(X, Y, W)
tau.hat = predict(tau.forest, X)
plot_uplift(tau.hat$predictions, W, Y, n_bs=20, x_interval = 0.05, balanced = FALSE)
plot_uplift(tau.hat$predictions, W, Y, n_bs=20, x_interval = 0.05, balanced = TRUE)
```

`plot_uplift_guelman` *A direct copy of Leo Guelman's uplift::qini function available in the R uplift package at commit 95965272e71c312623c95c439fb0b84f95c185b7: https://github.com/cran/uplift/blob/95965272e71c312623c95c439fb0b84f95c185b7/R/qini.R#L5*

Description

A direct copy of Leo Guelman's uplift::qini function available in the R uplift package at commit 95965272e71c312623c95c439fb0b84f95c185b7: https://github.com/cran/uplift/blob/95965272e71c312623c95c439fb0b84f

Usage

```
plot_uplift_guelman(p1, W, Y, p0 = rep(0, length(p1)), plotit = TRUE,
  direction = 1, groups = 10)
```

Arguments

`p1`	vector of numeric uplift predictions. Some uplift models produce two predictions: if-treated and if-control. In this case, if-treated predictions can be provided as p1, and if-control predictions can be provided as p0.
`W`	vector of 0,1 treatment indicators
`Y`	vector of 0,1 outcomes
`p0`	vector of numeric control predictions (default 0)
`plotit`	boolean plot the Qini chart
`direction`	1: calculate the differential response as p1-p0, 2: p0-p1
`groups`	5, 10, or 20: the number of quantiles in which to divide the population

Examples

```
set.seed(0)
rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p <- 3
beta <- -0.5
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.5)
Y <- rl(pmax(beta+X[,1], 0) * W + X[,2])
p1 <- 1/(1+exp(-(beta+X[,1])))
plot_uplift_guelman(p1, W, Y, groups=10, plotit=TRUE)


library(grf)
set.seed(123)

alpha <- 0.1
n <- 1000
W <- rbinom(n, 1, 0.5)
Y <- W
p1 <- Y + alpha*rnorm(n)
plot_uplift_guelman(p1, W, Y, groups=10)


rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p = 10
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.2)
Y <- rl(rl(X[,1]) * W - rl(X[,3]) * W + rnorm(n))
tau.forest <- causal_forest(X, Y, W)
tau.hat <- predict(tau.forest, X)
plot_uplift_guelman(tau.hat$predictions, W, Y)
```

`plUpliftEval` *A helper for the new_PlUpliftEval function that validates the treatment, outcome, prediction, p, and n_bins arguments.*

Description

A helper for the new_PlUpliftEval function that validates the treatment, outcome, prediction, p, and n_bins arguments.

Usage

```
plUpliftEval(treatment, outcome, prediction, p = "infer", n_bins = 20)
```

Arguments

treatment	numeric vector of treatment identifiers
outcome	numeric vector of outcomes
prediction	numeric vector of uplift predictions
p	optional "infer", numeric, numeric vector representing treatment propensities
n_bins	integer number of bins on x-axis; default 20

Value

a list representing a pylift uplift eval object

Examples

```
set.seed(0)
rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p <- 3
beta <- -0.5
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.5)
Y <- rl(pmax(beta+X[,1], 0) * W + X[,2])
p1 <- 1/(1+exp(-(beta+X[,1])))
plUpliftEval(W, Y, p1)


library(grf)
set.seed(123)

rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p <- 10
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.2)
Y <- rl(rl(X[,1]) * W - rl(X[,3]) * W + rnorm(n))
tau.forest <- causal_forest(X, Y, W)
tau.hat <- predict(tau.forest, X)
plue <- plUpliftEval(W, Y, tau.hat$predictions)
plue
```

pl_plot *A port of pylift's plot function (https://github.com/wayfair/pylift) as of commit: https://github.com/wayfair/pylift/tree/bb69692388b1fe085001c3ba7edf6dd81d888353 pylift: Plots the different kinds of percentage-targeted curves.*

Description

A port of pylift's plot function (https://github.com/wayfair/pylift) as of commit: https://github.com/wayfair/pylift/tree/bb6969
pylift: Plots the different kinds of percentage-targeted curves.

Usage

```
pl_plot(plue, plot_type = "cgains", n_bins = 20,
  show_theoretical_max = FALSE, show_practical_max = FALSE,
  show_random_selection = TRUE, show_no_dogs = FALSE, ...)
```

Arguments

plue	the result of a call to the plUpliftEval constructor
plot_type	string, optional Either 'qini', 'aqini', 'uplift', 'cuplift', or 'balance'. 'aqini' refers to an adjusted qini plot, 'cuplift' gives a cumulative uplift plot. 'balance' gives the test-control balance for each of the bins. All others are self-explanatory.
n_bins	integer, number of population bins; default 20
show_theoretical_max	boolean, optional Toggle theoretical maximal qini curve, if overfitting to treatment/control. Only works for Qini-style curves.
show_practical_max	boolean, optional Toggle theoretical maximal qini curve, if not overfitting to treatment/control. Only works for Qini-style curves.
show_random_selection	boolean, optional Toggle straight line indicating a random ordering. Only works for Qini-style curves.
show_no_dogs	boolean, optional Toggle theoretical maximal qini curve, if you believe there are no sleeping dogs. Only works for Qini-style curves.
...	additional arguments

Value

a pylift plot

Examples

```
set.seed(0)
rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p <- 3
beta <- -0.5
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.5)
Y <- rl(pmax(beta+X[,1], 0) * W + X[,2])
p1 <- 1/(1+exp(-(beta+X[,1])))
plue <- plUpliftEval(W, Y, p1)
pl_plot(plue,
        show_practical_max = TRUE,
        show_theoretical_max = TRUE,
        show_no_dogs = TRUE,
        n_bins=20)


library(grf)
set.seed(123)

rl <- function(x){
  round(1/(1+exp(-x)))
}
n <- 2000; p <- 10
X <- matrix(rnorm(n*p), n, p)
W <- rbinom(n, 1, 0.2)
Y <- rl(rl(X[,1]) * W - rl(X[,3]) * W + rnorm(n))
tau.forest <- causal_forest(X, Y, W)
tau.hat <- predict(tau.forest, X)
plue <- plUpliftEval(W, Y, tau.hat$predictions)
plue
pl_plot(plue,
        show_practical_max = TRUE,
        show_theoretical_max = TRUE,
        show_no_dogs = TRUE,
        n_bins=20)
```

Index