# Package 'uplift'

February 20, 2015

**Version** 0.3.5

**Date** 2014-03-17

**Title** Uplift Modeling

**Description** An integrated package for building and testing uplift models

**Author** Leo Guelman

**Maintainer** Leo Guelman <leo.guelman@gmail.com>

**License** GPL-2 | GPL-3

**Depends** R (>= 3.0.0), RItools, MASS, coin, tables, penalized

**Repository** CRAN

**NeedsCompilation** no

**Date/Publication** 2014-03-17 21:08:06

## R topics documented:

## uplift-package *Uplift Modeling*

### Description

An integrated package for building and testing uplift models.

### Details

| | |
|---|---|
| Package: | uplift |
| Type: | Package |
| Version: | 0.3.4 |
| Date: | 2014-01-03 |
| Depends: | R (>= 3.0.0), RItools, MASS, coin, tables, penalized |
| License: | GPL-2 \| GPL-3 |

### Author(s)

Leo Guelman <leo.guelman@gmail.com>

## ccif *Causal Conditional Inference Forest*

### Description

`ccif` implements recursive partitioning in a causal conditional inference framework.

### Usage

```
## S3 method for class 'formula'
ccif(formula, data, ...)

## Default S3 method:
ccif(
x,
y,
ct,
mtry = floor(sqrt(ncol(x))),
ntree = 100,
split_method = c("ED", "Chisq", "KL", "L1", "Int"),
```

```
interaction.depth = NULL,
pvalue = 0.05,
bonferroni = FALSE,
minsplit = 20,
minbucket_ct0 = round(minsplit/4),
minbucket_ct1 = round(minsplit/4),
keep.inbag = FALSE,
verbose = FALSE,
...)

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

## Arguments

| | |
|---|---|
| data | A data frame containing the variables in the model. It should include a variable reflecting the binary treatment assignment of each observation (coded as 0/1). |
| x, formula | a data frame of predictors or a formula describing the model to be fitted. A special term of the form `trt()` must be used in the model equation to identify the binary treatment variable. For example, if the treatment is represented by a variable named `treat`, then the right hand side of the formula must include the term `+trt(treat)`. |
| y | a binary response (numeric) vector. |
| ct | a binary (numeric) vector representing the treatment assignment (coded as 0/1). |
| mtry | the number of variables to be tested in each node; the default is floor(sqrt(ncol(x))). |
| ntree | the number of trees to generate in the forest; default is ntree = 100. |
| split_method | the split criteria used at each node of each tree; Possible values are: "ED" (Euclidean distance), "Chisq" (Chi-squared divergence), "KL" (Kullback-Leibler divergence), "Int" (Interaction method). |
| interaction.depth | The maximum depth of variable interactions. 1 implies an additive model, 2 implies a model with up to 2-way interactions, etc. |
| pvalue | the maximum acceptable pvalue required in order to make a split. |
| bonferroni | apply a bonferroni adjustment to pvalue. |
| minsplit | the minimum number of observations that must exist in a node in order for a split to be attempted. |
| minbucket_ct0 | the minimum number of control observations in any terminal <leaf> node. |
| minbucket_ct1 | the minimum number of treatment observations in any terminal <leaf> node. |
| keep.inbag | if set to `TRUE`, an nrow(x) by ntree matrix is returned, whose entries are the "in-bag" samples in each tree. |
| verbose | print status messages? |
| ... | Additional arguments passed to `independence_test{coin}`. See details. |

## Details

Causal conditional inference trees estimate *personalized treatment effects* (a.k.a. uplift) by binary recursive partitioning in a conditional inference framework. Roughly, the algorithm works as follows: 1) For each terminal node in the tree we test the global null hypothesis of no interaction effect between the treatment $T$ and any of the $n$ covariates selected at random from the set of $p$ covariates ($n \leq p$). Stop if this hypothesis cannot be rejected. Otherwise select the input variable with strongest interaction effect. The interaction effect is measured by a p-value corresponding to a permutation test (Strasser and Weber, 1999) for the partial null hypothesis of independence between each input variable and a transformed response. Specifically, the response is transformed so the impact of the input variable on the response has a causal interpretation for the treatment effect (see details in Guelman et al. 2013) 2) Implement a binary split in the selected input variable. 3) Recursively repeate steps 1) and 2).

The independence test between each input and the transformed response is performed by calling `independence_test{coin}`. Additional arguments may be passed to this function via '...'.

All split methods are described in Guelman et al. (2013a, 2013b).

This function is very slow at the moment. It was built as a prototype in R. A future version of this package will provide an interface to C++ for this function, which is expected to significantly improve speed.

## Value

An object of class `ccif`, which is a list with the following components:

| | |
|---|---|
| `call` | the original call to `ccif` |
| `trees` | the tree structure that was learned |
| `split_method` | the split criteria used at each node of each tree |
| `ntree` | the number of trees used |
| `mtry` | the number of variables tested at each node |
| `var.names` | a character vector with the name of the predictors |
| `var.class` | a character vector containing the class of each predictor variable |
| `inbag` | an nrow(x) by ntree matrix showing the in-bag samples used by each tree |

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013a). Uplift random forests. Cybernetics & Systems, *forthcoming*.

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013b). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

Hothorn, T., Hornik, K. and Zeileis, A. (2006).Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics, 15(3): 651-674.

Strasser, H. and Weber, C. (1999). On the asymptotic theory of permutation statistics. Mathematical Methods of Statistics, 8: 220-250.

## Examples

```
library(uplift)

### Simulate train data

set.seed(12345)
dd <- sim_pte(n = 100, p = 6, rho = 0, sigma =  sqrt(2), beta.den = 4)

dd$treat <- ifelse(dd$treat == 1, 1, 0)

### Fit model

form <- as.formula(paste('y ~', 'trt(treat) +', paste('X', 1:6, sep = '', collapse = "+")))

fit1 <- ccif(formula = form,
             data = dd,
             ntree = 50,
             split_method = "Int",
             distribution = approximate (B=999),
             pvalue = 0.05,
             verbose = TRUE)
print(fit1)
summary(fit1)
```

---

# checkBalance *Standardized Differences for Stratified Comparisons*

---

## Description

This function is simply a wrapper for `xBalance{RItools}`. Given covariates, a treatment variable, and (optionally) a stratifying factor, it calculates standardized mean differences along each covariate, and tests for conditional independence of the treatment variable and the covariates.

## Usage

```
checkBalance(formula, data, report = "all", ...)
```

## Arguments

| | |
|---|---|
| `formula` | A formula containing an indicator of treatment assignment on the left hand side and covariates at right. |
| `data` | A data frame in which the formula and (optionally) strata are to be evaluated. |
| `report` | Character vector listing measures to report for each stratification; a subset of c("adj.means","adj.mean.diffs", "adj.mean.diffs.null.sd", "chisquare.test", "std.diffs","z.scores","p.values' P-values reported are two-sided for the null-hypothesis of no effect. The option "all" requests all measures. |

... Additional arguments passed to xBalance{RItools}.

## Details

See help("xBalance") for details.

## Value

An object of class c("xbal", "list"). There are plot, print, and xtable methods for class xbal.

## Note

Evidence pertaining to the hypothesis that a treatment variable is not associated with differences in covariate values is assessed by comparing the differences of means (or regression coefficients), without standardization, to their distributions under hypothetical shuffles of the treatment variable, a permutation or randomization distribution. For the unstratified comparison, this reference distribution consists of differences (more generally, regression coefficients) when the treatment variable is permuted without regard to strata. For the stratified comparison, the reference distribution is determined by randomly permuting the treatment variable within strata, then re-calculating the treatment-control differences (regressions of each covariate on the permuted treatment variable). Significance assessments are based on the large-sample Normal approximation to these reference distributions.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Hansen, B.B. and Bowers, J. (2008). Covariate Balance in Simple, Stratified and Clustered Comparative Studies. Statistical Science 23.

Kalton, G. (1968). Standardization: A technique to control for extraneous variables. Applied Statistics 17, 118-136.

## Examples

```
library(uplift)

set.seed(12345)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

checkBalance(treat ~ X1 + X2 + X3 + X4 + X5 + X6 , data = dd)
```

`explore` *Explore Data for Uplift Modeling*

## Description

This function provides a basic exploratory tool for uplift modeling, by computing the average value of the response variable for each predictor and treatment assignment.

## Usage

```
explore(formula,
        data,
        subset,
        na.action = na.pass,
        nbins = 4,
        continuous = 4,
        direction = 1)
```

## Arguments

| | |
|---|---|
| `formula` | a formula expression of the form response ~ predictors. A special term of the form `trt()` must be used in the model equation to identify the binary treatment variable. For example, if the treatment is represented by a variable named `treat`, then the right hand side of the formula must include the term `+trt(treat)`. |
| `data` | a data.frame in which to interpret the variables named in the formula. |
| `subset` | expression indicating which subset of the rows of data should be included. All observations are included by default. |
| `na.action` | a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is `na.action = na.pass`. |
| `nbins` | the number of bins created from numeric predictors. The bins are created based on quantiles, with a default value of 4 (quartiles). |
| `continuous` | specifies the threshold for when a variable is considered to be continuous (when there are at least `continuous` unique values). The default is 4. Factor variables are always considered to be categorical no matter how many levels they have. |
| `direction` | possible values are `1` (default) if uplift should be computed as the difference in the average response between treatment and control, or `2` between control and treatment. This only affects the uplift calculation as produced in the output. |

## Value

A list of matrices, one for each variable. The columns represent: the number of responses over the control group, the number of the responses over the treated group, the average response for the control, the average response for the treatment, and the uplift (difference between treatment and control average response).

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## Examples

```
library(uplift)

set.seed(12345)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

eda <- explore(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat), data = dd)
```

---

`modelProfile` *Profile a Fitted Uplift Model*

---

## Description

This function can be used to profile a fitted uplift model. Given a vector of scores (uplift predictions), it computes basic summary statistics for each predictor by score quantile.

## Usage

```
modelProfile(formula, data, groups = 10,
group_label = c("I", "D"), digits_numeric = 1, digits_factor = 4,
exclude_na = FALSE, LaTex = FALSE)
```

## Arguments

| | |
|---|---|
| `formula` | a formula expression of the form score ~ predictors, where the LHS of the model formula should include the predictions from a fitted model. |
| `data` | a data.frame in which to interpret the variables named in the formula. |
| `groups` | number of groups of equal observations in which to partition the data set to show results. The default value is 10 (deciles). Other possible values are 5 and 20. |
| `group_label` | possible values are `"I"` or `"D"`, for group number labels which are increasing or decreasing with the model score, respectively. |
| `digits_numeric` | number of digits to show for numeric predictors. |
| `digits_factor` | number of digits to show for factor predictors. |
| `exclude_na` | should the results exclude observations with missing values in any of the variables named in the formula? |
| `LaTex` | should the function output LaTex code? |

## Details

This function ranks the variable supplied in the LHS of the model formula and classifies it into groups of equal number of observations. It subsequently calls the function `tabular` from the `tables` package to compute the average of each numeric predictor and the distribution of each factor within each group.

## Value

An object of S3 class `tabular`. See `help("tabular")` in the `tables` package for details.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## Examples

```
library(uplift)

### Simulate data
set.seed(12345)
dd <- sim_pte(n = 1000, p = 5, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0) # required coding for upliftRF

### Fit upliftRF model
fit1 <- upliftRF(y ~ X1 + X2 + X3 + X4 + X5 + trt(treat),
                 data = dd,
                 mtry = 3,
                 ntree = 50,
                 split_method = "KL",
                 minsplit = 100,
                 verbose = TRUE)

### Fitted values on train data
pred <- predict(fit1, dd)

### Compute uplift predictions
uplift_pred <- pred[, 1] - pred[, 2]

### Put together data, predictions and add some dummy factors for illustration only
dd2 <- data.frame(dd, uplift_pred, F1 = gl(2, 50, labels = c("A", "B")),
                  F2 = gl(4, 25, labels = c("a", "b", "c", "d")))

### Profile data based on fitted model
modelProfile(uplift_pred ~ X1 + X2 + X3 + F1 + F2,
             data = dd2,
             groups = 10,
             group_label = "D",
             digits_numeric = 2,
             digits_factor = 4,
             exclude_na = FALSE,
             LaTex = FALSE)
```

niv *Adjusted Net Information Value*

## Description

This function produces an adjusted net information value for each variable specified in the right hand side of the formula. This can be a helpful exploratory tool to (preliminary) determine the predictive power of each variable for uplift.

## Usage

```
niv(formula, data, subset, na.action = na.pass, B = 10, direction = 1,
nbins = 10, continuous = 4, plotit = TRUE, ...)
```

## Arguments

| | |
|---|---|
| formula | a formula expression of the form response ~ predictors. A special term of the form `trt()` must be used in the model equation to identify the binary treatment variable. For example, if the treatment is represented by a variable named `treat`, then the right hand side of the formula must include the term `+trt(treat)`. |
| data | a data.frame in which to interpret the variables named in the formula. |
| subset | expression indicating which subset of the rows of data should be included. All observations are included by default. |
| na.action | a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is `na.action = na.pass`. |
| B | the number of bootstrap samples used to compute the adjusted net information value. |
| direction | if set to 1 (default), the net weight of evidence is computed as the difference between the weight of evidence of the treatment and control groups, or if 2, it is computed as the difference between the weight of evidence of the control and treatment groups. This will not change the adjusted net information value, but only the sign of the net weight of evidence values. |
| nbins | the number of bins created from numeric predictors. The bins are created based on quantiles, with a default value of 10 (deciles). |
| continuous | specifies the threshold for when a variable is considered to be continuous (when there are at least `continuous` unique values). The default is 4. Factor variables are always considered to be categorical no matter how many levels they have. |
| plotit | plot the adjusted net information value for each variable? |
| ... | additional arguments passed to `barplot`. |

## Details

The ordinary information value (commonly used in credit scoring applications) is given by

$$IV = \sum_{i=1}^{G} (P(x = i|y = 1) - P(x = i|y = 0)) \times WOE_i$$

where $G$ is the number of groups created from a numeric predictor or categories from a categorical predictor, and $WOE_i = ln(\frac{P(x=i|y=1)}{P(x=i|y=0)})$.

The net information value is the natural extension of the IV for the case of uplift. It is computed as

$$NIV = 100 \times \sum_{i=1}^{G} (P(x = i|y = 1)^T \times P(x = i|y = 0)^C - P(x = i|y = 0)^T \times P(x = i|y = 1)^C) \times NWOE_i$$

where $NWOE_i = WOE_i^T - WOE_i^C$

The adjusted net information value is computed as follows:

1. Take $B$ bootstrap samples and compute the NIV for each variable on each sample

2. Compute the mean of the NIV ($NIV_{mean}$) and sd of the NIV ($NIV_{sd}$) for each variable over all the $B$ bootstraps

3. The adjusted NIV for a given variable is computed by adding a penalty term to the mean NIV: $NIV_{mean} - \frac{NIV_{sd}}{\sqrt{B}}$.

## Value

A list with two components:

`niv_val` a matrix with the following columns: niv (the average net information value for each variable over all bootstrap samples), penalty (the penalty term calculated as described in the details above), the adjusted information value (the difference between the prior two colums)

`nwoe` a list of matrices, one for each variable. The columns represent: the distribution of the responses (y=1) over the treated group (`ct1.y1`), the distribution of the non-responses (y=0) over the treated group (`ct1.y0`), the distribution of the responses (y=1) over the control group (`ct0.y1`), the distribution of the non-responses (y=0) over the control group (`ct0.y0`), the weight-of-evidence over the treated group (`ct1.woe`), the weight-of-evidence over the control group `ct0.woe`, and the net weigh-of-evidence (`nwoe`).

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Larsen, K. (2009). Net lift models. In: M2009 - 12th Annual SAS Data Mining Conference.

## Examples

```
library(uplift)

set.seed(12345)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

niv.1 <- niv(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat), data = dd)
niv.1$niv
niv.1$nwoe
```

---

# performance *Performance Assessment for Uplift Models*

---

## Description

Provides a method for assessing performance for uplift models.

## Usage

```
performance(pr.y1_ct1, pr.y1_ct0, y, ct, direction = 1, groups = 10)
```

## Arguments

| | |
|---|---|
| pr.y1_ct1 | the predicted probability $Prob(y = 1\|treated, x)$. |
| pr.y1_ct0 | the predicted probability $Prob(y = 1\|control, x)$. |
| y | the actual observed value of the response. |
| ct | a binary (numeric) vector representing the treatment assignment (coded as 0/1). |
| direction | possible values are 1 (default) if the objective is to maximize the difference in the response for Treatment minus Control, and 2 for Control minus Treatment. |
| groups | number of groups of equal observations in which to partition the data set to show results. The default value is 10 (deciles). Other possible values are 5 and 20. |

## Details

Model performance is estimated by: 1. computing the difference in the predicted conditional class probabilities $Prob(y = 1|treated, x)$ and $Prob(y = 1|control, x)$, 2. ranking the difference and grouping it into 'buckets' with equal number of observations each, and 3. computing the actual difference in the mean of the response variable between the treatment and the control groups for each bucket.

## Value

An object of class `performance`, which is a matrix with the following columns: (`group`) the number of groups, (`n.ct1`) the number of observations in the treated group, (`n.ct0`) the number of observations in the control group, (`n.y1_ct1`) the number of observation in the treated group with response = 1, (`n.y1_ct0`) the number of observation in the control group with response = 1, (`r.y1_ct1`) the mean of the response for the treated group, (`r.y1_ct0`) the mean of the response for the control group, and (`uplift`) the difference between `r.y1_ct1` and `r.y1_ct0` (if `direction = 1`).

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Uplift random forests. Cybernetics & Systems, *forthcoming*.

## Examples

```
library(uplift)

set.seed(123)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### fit uplift random forest

fit1 <- upliftRF(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat),
                 data = dd,
                 mtry = 3,
                 ntree = 100,
                 split_method = "KL",
             minsplit = 200, # need small trees as there is strong uplift effects in the data
                 verbose = TRUE)
print(fit1)
summary(fit1)

### get variable importance

varImportance(fit1, plotit = TRUE, normalize = TRUE)

### predict on new data

dd_new <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd_new$treat <- ifelse(dd_new$treat == 1, 1, 0)

pred <- predict(fit1, dd_new)

### evaluate model performance

perf <- performance(pred[, 1], pred[, 2], dd_new$y, dd_new$treat, direction = 1)
```

```
plot(perf[, 8] ~ perf[, 1], type ="l", xlab = "Decile", ylab = "uplift")
```

---

predict.ccif *Predictions from a Fitted Causal Conditional Inference Forest Model*

---

## Description

prediction of new data using causal conditional inference forest.

## Usage

```
## S3 method for class 'ccif'
predict(object, newdata, n.trees = object$ntree, predict.all = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | an object of class ccif, as that created by the function ccif. |
| newdata | a data frame containing the values at which predictions are required. |
| n.trees | number of trees used in the prediction; The default is object$ntree. |
| predict.all | should the predictions of all trees be kept? |
| ... | not used. |

## Details

At the moment, all predictors passed for fitting the uplift model must also be present in newdata, even if they are not used as split variables by any of the trees in the forest.

## Value

If predict.all = FALSE, a matrix of predictions containing the conditional class probabilities: pr.y1_ct1 represents $Prob(y = 1|treated, x)$ and pr.y1_ct0 represents $Prob(y = 1|control, x)$. This is computed as the average of the individual predictions over all trees.

If predict.all = TRUE, the returned object is a list with two components: pred.avg is the prediction (as described above) and individual is a list of matrices containing the individual predictions from each tree.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

## Examples

```
library(uplift)

### Simulate train data

set.seed(12345)
dd <- sim_pte(n = 100, p = 6, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### Fit model

form <- as.formula(paste('y ~', 'trt(treat) +', paste('X', 1:6, sep = '', collapse = "+")))

fit1 <- ccif(formula = form,
             data = dd,
             ntree = 50,
             split_method = "Int",
             pvalue = 0.05,
             verbose = TRUE)

### Predict on new data

dd_new <- sim_pte(n = 200, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)

pred <- predict(fit1, dd_new)
```

---

# predict.upliftRF *Predictions from a Fitted Uplift Random Forest Model*

---

## Description

prediction of new data using uplift random forest.

## Usage

```
## S3 method for class 'upliftRF'
predict(object, newdata, n.trees = object$ntree, predict.all = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | an object of class upliftRF, as that created by the function upliftRF. |
| newdata | a data frame containing the values at which predictions are required. |
| n.trees | number of trees used in the prediction; The default is object$ntree. |
| predict.all | should the predictions of all trees be kept? |
| ... | not used. |

## Details

At the moment, all predictors passed for fitting the uplift model must also be present in `newdata`, even if they are not used as split variables by any of the trees in the forest.

## Value

If `predict.all = FALSE`, a matrix of predictions containing the conditional class probabilities: `pr.y1_ct1` represents $Prob(y = 1|treated, x)$ and `pr.y1_ct0` represents $Prob(y = 1|control, x)$. This is computed as the average of the individual predictions over all trees.

If `predict.all = TRUE`, the returned object is a list with two components: `pred.avg` is the prediction (as described above) and `individual` is a list of matrices containing the individual predictions from each tree.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Uplift random forests. Cybernetics & Systems, *forthcoming*.

## See Also

upliftRF

## Examples

```
library(uplift)

### simulate data for uplift modeling

set.seed(123)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### fit uplift random forest

fit1 <- upliftRF(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat),
                 data = dd,
                 mtry = 3,
                 ntree = 100,
                 split_method = "KL",
              minsplit = 200, # need small trees as there is strong uplift effects in the data
                 verbose = TRUE)
summary(fit1)

### predict on new data

dd_new <- sim_pte(n = 2000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
```

```
dd_new$treat <- ifelse(dd_new$treat == 1, 1, 0)

pred <- predict(fit1, dd_new)
head(pred)
```

---

qini *Computes the Qini Coefficient Q*

---

## Description

This function computes the Qini coefficient from a `performance` object (as created by the function `performance`).

## Usage

```
## S3 method for class 'performance'
qini(x, direction = 1, plotit = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object of class `performance`. |
| `direction` | possible values are `1` (default) if the objective is to maximize the difference in the response for Treatment minus Control, and `2` for Control minus Treatment. |
| `plotit` | plot the incremental gains from the fitted model? |
| `...` | additional arguments passed to `plot`. |

## Details

Qini coefficients represent a natural generalizations of the Gini coefficient to the case of uplift. Qini is defined as the area between the actual incremental gains curve from the fitted model and the area under the diagonal corresponding to random targeting. See the references for details.

## Value

A list with the following components

| | |
|---|---|
| `Qini` | the Qini coefficient as defined above. |
| `inc.gains` | the incremental gain values from the fitted model. |
| `random.inc.gains` | the random incremental gains. |

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Radcliffe, N. and Surry, P. (2011). Real-World Uplift Modelling with Significance-Based Uplift Trees. Portrait Technical Report, TR-2011-1.

Radcliffe, N. (2007). Using control groups to target on predicted lift: Building and assessing uplift models. Direct Marketing Analytics Journal, An Annual Publication from the Direct Marketing Association Analytics Council, pages 14-21.

## Examples

```
library(uplift)

### simulate data for uplift modeling

set.seed(123)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### fit uplift random forest

fit1 <- upliftRF(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat),
                 data = dd,
                 mtry = 3,
                 ntree = 100,
                 split_method = "KL",
              minsplit = 200, # need small trees as there is strong uplift effects in the data
                 verbose = TRUE)
print(fit1)
summary(fit1)

### predict on new data

dd_new <- sim_pte(n = 2000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd_new$treat <- ifelse(dd_new$treat == 1, 1, 0)

pred <- predict(fit1, dd_new)

### evaluate model performance

perf <- performance(pred[, 1], pred[, 2], dd_new$y, dd_new$treat, direction = 1)

### compute Qini coefficient

Q <- qini(perf, plotit = TRUE)
Q
```

---

rvtu *Response Variable Transform for Uplift Modeling*

---

## Description

This function transforms the data frame supplied in the function call by creating a new response variable and an equal number of control and treated observations. This transformed data set can be subsequently used with any conventional supervised learning algorithm to model uplift.

## Usage

```
rvtu(formula, data, subset, na.action = na.pass,
     method = c("undersample", "oversample", "weights", "none"))
```

## Arguments

| | |
|---|---|
| `formula` | a formula expression of the form response ~ predictors. A special term of the form `trt()` must be used in the model equation to identify the binary treatment variable. For example, if the treatment is represented by a variable named `treat`, then the right hand side of the formula must include the term `+trt(treat)`. |
| `data` | a data.frame in which to interpret the variables named in the formula. |
| `subset` | expression indicating which subset of the rows of data should be included. All observations are included by default. |
| `na.action` | a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is `na.action = na.pass`. |
| `method` | the method used to create the transformed data set. It must be one of "undersample", "oversample", "weights" or "none", with no default. See details. |

## Details

The transformed response variable $z$ equals 1 if the observation has a response value of 1 and has been treated, or if it has a response value of 0 and has not been treated. Intuitively, $z$ equals 1 if we know that, for a given case, the outcome in the treatment group would have been at least as good as in the control group, had we known for this case the outcome in both groups. Under equal proportion of control and treated observations, it is easy to prove that $2 * Prob(z = 1|x) - 1 = Prob(y = 1|treated, x) - Prob(y = 1|control, x)$ (Jaskowski and Jaroszewicz, 2012).

If the data has an equal number of control and treated observations, then `method = "none"` must be used. Otherwise, any of the other methods must be used.

If `method = "undersample"`, a random sample without replacement is drawn from the treated class (i.e., treated/control) with the majority of observations, such that the returned data frame will have balanced treated/control proportions.

If `method = "oversample"`, a random sample with replacement is drawn from the treated class with the minority of observations, such that the returned data frame will have balanced treated/control proportions.

If `method = "weights"`, the returned data frame will have a weight variable $w$ assigned to each observation. The weight assigned to the treated (control) equals 1 - proportion of treated observations (proportion of treated observations).

## Value

A data frame including the predictor variables (RHS of the formula expression), the treatment ($ct = 1$) and control ($ct = 0$) assignment, the original response variable (LHS of the formula expression), and the transformed response variable for uplift modeling $z$. If `method = "weights"` an additional weight variable $w$ is included.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Jaskowski, M. and Jaroszewicz, S. (2012) Uplift Modeling for Clinical Trial Data. In ICML 2012 Workshop on Machine Learning for Clinical Data Analysis, Edinburgh, Scotland.

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

## Examples

```
library(uplift)

### Simulate data

set.seed(1)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### Transform response variable for uplift modeling
dd2 <- rvtu(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat), data = dd, method = "none")

### Fit a Logistic model to the transformed response
glm.uplift <- glm(z ~ X1 + X2 + X3 + X4 + X5 + X6, data = dd2, family = "binomial")

### Test fitted model on new data
dd_new <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd_new$treat <- ifelse(dd_new$treat == 1, 1, 0)
pred <- predict(glm.uplift, dd_new, type = "response")
perf <- performance(2 * pred - 1, rep(0, length(pred)), dd_new$y, dd_new$treat, direction = 1)
perf
```

---

`sim_pte` *Simulations for Personalized Treatment Effects*

---

## Description

Numerical simulation for treatment effect heterogeneity estimation as described in Tian et al. (2012)

## Usage

```
sim_pte(n = 1000, p = 20, rho = 0, sigma = sqrt(2), beta.den = 4)
```

## Arguments

| | |
|---|---|
| n | number of observations. |
| p | number of predictors. |
| rho | covariance between predictors. |
| sigma | multiplier of error term. |
| beta.den | size of main effects relative to interaction effects. See details. |

## Details

sim_pte simulates data according to the following specification:

$$Y = I(\sum_{j=1}^{p} \beta_j X_j + \sum_{j=1}^{p} \gamma_j X_j T + \sigma_0 \epsilon > 0)$$

,

where $\gamma = (1/2, -1/2, 1/2, -1/2, 0, ..., 0)$, $\beta = (-1)^{j+1} I(3 \leq j \leq 10)/$beta.den, $(X_1, \ldots, X_p)$ follows a mean zero multivariate normal distribution with a compound symmetric variance-covariance matrix, $(1-\rho)\mathbf{I}_p + \rho \mathbf{1}^T \mathbf{1}$, $T = [-1, 1]$ is the treatment indicator and $\epsilon$ is $N(0, 1)$.

In this case, the "true" treatment effect score $(Prob(Y = 1|T = 1) - Prob(Y = 1|T = -1))$ is given by

$$\Phi(\frac{\sum_{j=1}^{p}(\beta_j + \gamma_j)X_j}{\sigma_0}) - \Phi(\frac{\sum_{j=1}^{p}(\beta_j - \gamma_j)X_j}{\sigma_0})$$

.

## Value

A data frame including the response variable ($Y$), the treatment (treat=1) and control (treat=-1) assignment, the predictor variables ($X$) and the "true" treatment effect score (ts)

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Tian, L., Alizadeh, A., Gentles, A. and Tibshirani, R. 2012. A simple method for detecting interactions between a treatment and a large number of covariates. Submitted on Dec 2012. arXiv:1212.2995 [stat.ME].

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

## Examples

```
library(uplift)
### Simulate train data

set.seed(12345)
dd <- sim_pte(n = 1000, p = 10, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0) # required coding for upliftRF

### Fit model

form <- as.formula(paste('y ~', 'trt(treat) +', paste('X', 1:10, sep = '', collapse = "+")))

fit1 <- upliftRF(formula = form,
                 data = dd,
                 ntree = 100,
                 split_method = "Int",
                 interaction.depth = 3,
                 minsplit = 100,
                 minbucket_ct0 = 50,
                 minbucket_ct1 = 50,
                 verbose = TRUE)
summary(fit1)
```

---

# tian_transf *Modify Covariates for Uplift Modeling*

---

## Description

This function transforms the data frame supplied in the function call by creating a new set of modified covariates and an equal number of control and treated observations. This transformed data set can be subsequently used with any conventional supervised learning algorithm to model uplift.

## Usage

```
tian_transf(formula, data, subset, na.action = na.pass,
 method = c("undersample", "oversample", "none"),
 standardize = TRUE, cts = FALSE)
```

## Arguments

| | |
|---|---|
| formula | a formula expression of the form response ~ predictors. A special term of the form `trt()` must be used in the model equation to identify the binary treatment variable. For example, if the treatment is represented by a variable named `treat`, then the right hand side of the formula must include the term `+trt(treat)`. |
| data | a data.frame in which to interpret the variables named in the formula. |
| subset | expression indicating which subset of the rows of data should be included. All observations are included by default. |

| | |
|---|---|
| na.action | a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is na.action = na.pass. |
| method | the method used to create the transformed data set. It must be one of "undersample", "oversample" or "none", with no default. See details. |
| standardize | If TRUE, each variable is standardized to have unit L2 norm, otherwise it is left alone. Default is TRUE. |
| cts | if TRUE, contrasts for factors are created in a special way. See details. Default is FALSE. |

## Details

The covariates $x$ supplied in the RHS of the model formula are transformed as $w = z * T/2$, where $T = [-1, 1]$ is the treatment indicator and $z$ is the matrix of standardize $x$ variables.

If cts = TRUE, factors included in the formula are converted to dummy variables in a special way that is more appropriate when the returned model frame is used to fit a penalized regression. In this case, contrasts used for factors are given by penalized regression contrasts from the penalized package. Unordered factors are turned into as many dummy variables as the factor has levels, except when the number of levels is 2, in which case it returns a single contrast. This ensures a symmetric treatment of all levels and guarantees that the fit does not depend on the ordering of the levels. See help(contr.none) in penalized package. Ordered factors are turned into dummy variables that code for the difference between successive levels (one dummy less than the number of levels). See help(contr.diff) in penalized package.

If the data has an equal number of control and treated observations, then method = "none" should be used. Otherwise, any of the other methods should be used.

If method = "undersample", a random sample without replacement is drawn from the treated class (i.e., treated/control) with the majority of observations, such that the returned data frame will have balanced treated/control proportions.

If method = "oversample", a random sample with replacement is drawn from the treated class with the minority of observations, such that the returned data frame will have balanced treated/control proportions.

## Value

A model frame, including the modified covariates $w$ (the prefix "T_" is added to the name of each covariate to denote it has been modified), the treatment ($ct = 1$) and control ($ct = 0$) assignment and the response variable (LHS of model formula). The intercept is omitted from the model frame.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Tian, L., Alizadeh, A., Gentles, A. and Tibshirani, R. 2012. A simple method for detecting interactions between a treatment and a large number of covariates. Submitted on Dec 2012. arXiv:1212.2995 [stat.ME].

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

## Examples

```
library(uplift)

set.seed(1)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

dd2 <- tian_transf(y ~  X1 + X2 + X3 + trt(treat), data =dd, method = "none")
head(dd2)
```

---

# trt *Mark Treatment Term*

---

## Description

This is a dummy function, used to mark the treatment term in various functions within the `uplift` package.

## Usage

```
trt(x)
```

## Arguments

| | |
|---|---|
| x | A numeric variable coded as 1 (treatment) and 0 (control). |

## Value

x, unchanged

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

upliftKNN *Uplift k-Nearest Neighbor*

## Description

upliftKNN implements k-nearest neighbor for uplift modeling.

## Usage

```
upliftKNN(train, test, y, ct, k = 1, dist.method = "euclidean",
          p = 2, ties.meth = "min", agg.method = "mean")
```

## Arguments

| | |
|---|---|
| train | a matrix or data frame of training set cases. |
| test | a matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case. |
| y | a numeric response variable (must be coded as 0/1 for binary response). |
| ct | factor or numeric vector representing the treatment to which each train case is assigned. At least 2 groups are required (e.g. treatment and control). Multi-treatments are also supported. |
| k | number of neighbors considered. |
| dist.method | the distance to be used in calculating the neighbors. Any method supported in function dist is valid. |
| p | the power of the Minkowski distance. |
| ties.meth | method to handle ties for the kth neighbor. The default is "min" which uses all ties. Alternatives include "max" which uses none if there are ties for the k-th nearest neighbor, "random" which selects among the ties randomly and "first" which uses the ties in their order in the data. |
| agg.method | method to combine responses of the nearest neighbors, defaults to "mean". The alternative is "majority". |

## Details

k-nearest neighbor for uplift modeling for a test set from a training set. For each case in the test set, the k-nearest training set vectors for each treatment type are found. The response value for the k-nearest training vectors is aggregated based on the function specified in agg.method. For "majority", classification is decided by majority vote (with ties broken at random).

## Value

A matrix of predictions for each test case and value of ct

## Note

The code logic follows closely the knn and knnflex packages, the later currently discontinued from CRAN.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Su, X., Kang, J., Fan, J., Levine, R. A., and Yan, X. (2012). Facilitating score and causal inference trees for large observational studies. Journal of Machine Learning Research, 13(10): 2955-2994.

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

## Examples

```
library(uplift)

### simulate data for uplift modeling

set.seed(1)

train <- sim_pte(n = 500, p = 10, rho = 0, sigma =  sqrt(2), beta.den = 4)
train$treat <- ifelse(train$treat == 1, 1, 0)

### Fit an Uplift k-Nearest Neighbor on test data

test <- sim_pte(n = 100, p = 10, rho = 0, sigma =  sqrt(2), beta.den = 4)
test$treat <- ifelse(test$treat == 1, 1, 0)

fit1 <- upliftKNN(train[, 3:8], test[, 3:8], train$y, train$treat, k = 1,
         dist.method = "euclidean", p = 2, ties.meth = "min",   agg.method = "majority")
head(fit1)
```

---

# upliftRF *Uplift Random Forests*

---

## Description

upliftRF implements Random Forests with split criteria designed for binary uplift modeling tasks.

## Usage

```
## S3 method for class 'formula'
upliftRF(formula, data, ...)

## Default S3 method:
upliftRF(
x,
y,
ct,
mtry = floor(sqrt(ncol(x))),
ntree = 100,
split_method = c("ED", "Chisq", "KL", "L1", "Int"),
interaction.depth = NULL,
bag.fraction = 0.5,
minsplit = 20,
minbucket_ct0 = round(minsplit/4),
minbucket_ct1 = round(minsplit/4),
keep.inbag = FALSE,
verbose = FALSE,
...)

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

## Arguments

| | |
|---|---|
| data | A data frame containing the variables in the model. It should include a variable reflecting the binary treatment assignment of each observation (coded as 0/1). |
| x, formula | a data frame of predictors or a formula describing the model to be fitted. A special term of the form `trt()` must be used in the model equation to identify the binary treatment variable. For example, if the treatment is represented by a variable named `treat`, then the right hand side of the formula must include the term `+trt(treat)`. |
| y | a binary response (numeric) vector. |
| ct | a binary (numeric) vector representing the treatment assignment (coded as 0/1). |
| mtry | the number of variables to be tested in each node; the default is floor(sqrt(ncol(x))). |
| ntree | the number of trees to generate in the forest; default is ntree = 100. |
| split_method | the split criteria used at each node of each tree; Possible values are: "ED" (Euclidean distance), "Chisq" (Chi-squared divergence), "KL" (Kullback-Leibler divergence), "Int" (Interaction method). |
| interaction.depth | The maximum depth of variable interactions. 1 implies an additive model, 2 implies a model with up to 2-way interactions, etc. The default is to grow trees to maximal depth, constrained on the arguments specified in `minsplit` and `minbucket`. |

| | |
|---|---|
| bag.fraction | the fraction of the training set observations randomly selected for the purpose of fitting each tree in the forest. |
| minsplit | the minimum number of observations that must exist in a node in order for a split to be attempted. |
| minbucket_ct0 | the minimum number of control observations in any terminal <leaf> node. |
| minbucket_ct1 | the minimum number of treatment observations in any terminal <leaf> node. |
| keep.inbag | if set to TRUE, an nrow(x) by ntree matrix is returned, whose entries are the "in-bag" samples in each tree. |
| verbose | print status messages? |
| ... | optional parameters to be passed to the low level function upliftRF.default. |

## Details

Uplift Random Forests estimate *personalized treatment effects* (a.k.a. uplift) by binary recursive partitioning. The algorithm and split methods are described in Guelman et al. (2013a, 2013b).

## Value

An object of class upliftRF, which is a list with the following components:

| | |
|---|---|
| call | the original call to upliftRF |
| trees | the tree structure that was learned |
| split_method | the split criteria used at each node of each tree |
| ntree | the number of trees used |
| mtry | the number of variables tested at each node |
| var.names | a character vector with the name of the predictors |
| var.class | a character vector containing the class of each predictor variable |
| inbag | an nrow(x) by ntree matrix showing the in-bag samples used by each tree |

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013a). Uplift random forests. Cybernetics & Systems, *forthcoming*.

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013b). Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. *Submitted*.

Su, X., Tsai, C., Wang, H., Nickerson, D., and Li, B. (2009). Subgroup Analysis via Recursive Partitioning. Journal of Machine Learning Research, 10, 141-158.

## Examples

```
library(uplift)

### simulate data for uplift modeling

set.seed(123)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### fit uplift random forest

fit1 <- upliftRF(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat),
                 data = dd,
                 mtry = 3,
                 ntree = 100,
                 split_method = "KL",
                 minsplit = 200,
                 verbose = TRUE)
print(fit1)
summary(fit1)
```

---

# varImportance *Extract Variable Importance from* upliftRF *or* ccif *Fitted Objects*

---

## Description

This is the extractor function for variable importance of predictors.

## Usage

```
## S3 method for class 'upliftRF'
varImportance(x, n.trees = x$ntree, plotit = TRUE, normalize = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | an object of class upliftRF or ccif. |
| n.trees | number of trees used in the prediction; The default is x$ntree. |
| plotit | plot variable importance? |
| normalize | if set to TRUE, the importance is scaled to add up to 100. |
| ... | additional arguments passed to barplot. |

## Details

At each split in each tree, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable.

## Value

A numeric vector with the variable importance.

## Author(s)

Leo Guelman <leo.guelman@gmail.com>

## References

Guelman, L., Guillen, M., and Perez-Marin A.M. (2013). Uplift random forests. Cybernetics & Systems, *forthcoming*.

## Examples

```
library(uplift)

### simulate data for uplift modeling

set.seed(123)
dd <- sim_pte(n = 1000, p = 20, rho = 0, sigma =  sqrt(2), beta.den = 4)
dd$treat <- ifelse(dd$treat == 1, 1, 0)

### fit uplift random forest

fit1 <- upliftRF(y ~ X1 + X2 + X3 + X4 + X5 + X6 + trt(treat),
                 data = dd,
                 mtry = 3,
                 ntree = 100,
                 split_method = "KL",
                 minsplit = 200,
                 verbose = TRUE)
print(fit1)

### get variable importance

varImportance(fit1, plotit = TRUE, normalize = TRUE)
```

# Index