# Package 'xrf'

October 14, 2022

**Title** eXtreme RuleFit

**Version** 0.2.2

**Description** An implementation of the RuleFit algorithm as described in Friedman & Popescu
(2008) <doi:10.1214/07-AOAS148>. eXtreme Gradient Boosting ('XGBoost') is used
to build rules, and 'glmnet' is used to fit a sparse linear model on the raw and rule fea-
tures. The result
is a model that learns similarly to a tree ensemble, while often offering improved interpretability
and achieving improved scoring runtime in live applications. Several algorithms for
reducing rule complexity are provided, most notably hyperrectangle de-
overlapping. All algorithms scale to
several million rows and support sparse representations to handle tens of thousands of dimensions.

**URL** https://github.com/holub008/xrf

**BugReports** https://github.com/holub008/xrf/issues

**Depends** R (>= 3.1.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Imports** Matrix, glmnet (>= 3.0), xgboost (>= 0.71.2), dplyr,
fuzzyjoin, rlang, methods

**Suggests** testthat, covr

**NeedsCompilation** no

**Author** Karl Holub [aut, cre]

**Maintainer** Karl Holub <karljholub@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-10-04 06:00:02 UTC

## R topics documented:

---

coef.xrf                      *Produce rules & coefficients for the RuleFit model*

---

### Description

Produce rules & coefficients for the RuleFit model

### Usage

```
## S3 method for class 'xrf'
coef(object, lambda = "lambda.min", ...)
```

### Arguments

| | |
|---|---|
| object | an object of class "xrf" |
| lambda | the lasso penalty parameter to be applied as in 'glmnet' |
| ... | ignored arguments |

### Examples

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
linear_model_coefficients <- coef(m, lambda = 'lambda.1se')
```

---

model.matrix.xrf         *Generate the design matrix from an eXtreme RuleFit model*

---

### Description

Generate the design matrix from an eXtreme RuleFit model

### Usage

```
## S3 method for class 'xrf'
model.matrix(object, data, sparse = TRUE, ...)
```

## Arguments

| | |
|---|---|
| object | an object of class "xrf" |
| data | data to generate design matrix from |
| sparse | a logical indicating whether a sparse design matrix should be used |
| ... | ignored arguments |

## Examples

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
design <- model.matrix(m, iris, sparse = FALSE)
```

---

| predict.xrf | *Draw predictions from a RuleFit xrf model* |
|---|---|

---

## Description

Draw predictions from a RuleFit xrf model

## Usage

```
## S3 method for class 'xrf'
predict(
  object,
  newdata,
  sparse = TRUE,
  lambda = "lambda.min",
  type = "response",
  ...
)
```

## Arguments

| | |
|---|---|
| object | an object of class "xrf" |
| newdata | data to predict on |
| sparse | a logical indicating whether a sparse design matrix should be used |
| lambda | the lasso penalty parameter to be applied |
| type | the type of predicted value produced |
| ... | ignored arguments |

**Examples**

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
predictions <- predict(m, iris)
```

---

print.xrf                    *Print an eXtreme RuleFit model*

---

**Description**

Print an eXtreme RuleFit model

**Usage**

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

**Arguments**

x               an object of class "xrf"

...             ignored arguments

**Examples**

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
print(m)
```

---

summary.xrf                  *Summarize an eXtreme RuleFit model*

---

**Description**

Summarize an eXtreme RuleFit model

**Usage**

```
## S3 method for class 'xrf'
summary(object, ...)
```

## Arguments

| object | an object of class "xrf" |
|---|---|
| ... | ignored arguments |

## Examples

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
summary(m)
```

---

| xrf | *Fit an eXtreme RuleFit model* |
|---|---|

---

## Description

S3 method for building an "eXtreme RuleFit" model. See `xrf.formula` for preferred entry point

## Usage

```
xrf(object, ...)
```

## Arguments

| object | an object describing the model to be fit |
|---|---|
| ... | additional arguments |

## Examples

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
```

---

| xrf.formula | *Fit an eXtreme RuleFit model* |
|---|---|

---

## Description

See Friedman & Popescu (2008) for a description of the general RuleFit algorithm. This method uses XGBoost to fit a tree ensemble, extracts a ruleset as the conjunction of tree traversals, and fits a sparse linear model to the resulting feature set (including the original feature set) using glmnet.

**Usage**

```
## S3 method for class 'formula'
xrf(
  object,
  data,
  family,
  xgb_control = list(nrounds = 100, max_depth = 3),
  glm_control = list(type.measure = "deviance", nfolds = 5),
  sparse = TRUE,
  prefit_xgb = NULL,
  deoverlap = FALSE,
  ...
)
```

**Arguments**

| | |
|---|---|
| `object` | a formula prescribing features to use in the model. transformation of the response variable is not supported. when using transformations on the input features (not suggested in general) it is suggested to set sparse=F |
| `data` | a data frame with columns corresponding to the formula |
| `family` | the family of the fitted model. one of 'gaussian', 'binomial', 'multinomial' |
| `xgb_control` | a list of parameters for xgboost. must supply an nrounds argument |
| `glm_control` | a list of parameters for the glmnet fit. must supply a type.measure and nfolds arguments (for the lambda cv) |
| `sparse` | whether a sparse design matrix should be used |
| `prefit_xgb` | an xgboost model (of class xgb.Booster) to be used instead of the model that `xrf` would normally fit |
| `deoverlap` | if true, the tree derived rules are deoverlapped, in that the deoverlapped rule set contains no overlapped rules |
| `...` | ignored arguments |

**References**

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

**Examples**

```
m <- xrf(Petal.Length ~ ., iris,
         xgb_control = list(nrounds = 2, max_depth = 2),
         family = 'gaussian')
```

# Index