Package 'r2pmml'

January 24, 2025

Version 0.30.0

Date 2025-01-25

Type Package

License AGPL-3

Title Convert R Models to PMML

Description R wrapper for the JPMML-R library <https://github.com/jpmml/jpmml-r>, which converts R models to Predictive Model Markup Language (PMML).

URL https://github.com/jpmml/r2pmml

LazyLoad yes

NeedsCompilation yes

RoxygenNote 7.2.1

Imports methods

Suggests caret, e1071, earth, evtree, glmnet, lightgbm, mlbench, mlr, partykit, randomForest, ranger, xgboost

SystemRequirements Java (>= 8.0)

Author Villu Ruusmann [aut, cre]

Maintainer Villu Ruusmann <villu.ruusmann@gmail.com>

Repository CRAN

Date/Publication 2025-01-24 18:20:02 UTC

Contents

`as.fmap` *Dispatches execution to the most appropriate XGBoost feature map generation function.*

Description

Dispatches execution to the most appropriate XGBoost feature map generation function.

Usage

```
as.fmap(x)
```

Arguments

`x`	A dataset object.

`as.fmap.data.frame` *Generates an XGBoost feature map based on feature data.*

Description

Generates an XGBoost feature map based on feature data.

Usage

```
## S3 method for class 'data.frame'
as.fmap(x)
```

Arguments

x	A "data.frame" object with independent variables.

Value

A "data.frame" object.

Examples

```
data(iris)
iris.df = iris[, 1:4]
iris.fmap = as.fmap(iris.df)
```

---

as.fmap.matrix *Generates an XGBoost feature map based on feature data.*

Description

Generates an XGBoost feature map based on feature data.

Usage

```
## S3 method for class 'matrix'
as.fmap(x)
```

Arguments

x	A "matrix" object with independent variables.

Value

A "data.frame" object.

Examples

```
data(iris)
iris.matrix = model.matrix(Species ~ . - 1, data = iris)
iris.fmap = as.fmap(iris.matrix)
```

`as.scorecard` *Converts a "glm" object to a "scorecard" object.*

Description

Converts a "glm" object to a "scorecard" object.

Usage

```
as.scorecard(glm, odds = 10, base_points = 500, pdo = 100)
```

Arguments

`glm`	A "glm" object with binomial family link function.
`odds`	Odds ratio at base odds.
`base_points`	Points where odds ratio is defined.
`pdo`	Points to double the odds.

Value

A "scorecard" object.

`decorate` *Dispatches execution to the most appropriate model decoration function.*

Description

Dispatches execution to the most appropriate model decoration function.

Usage

```
decorate(x, ...)
```

Arguments

`x`	A model object.
`...`	Arguments to pass on to the selected function.

`decorate.default` *Decorates a model object with "preProcess" and "pmml_options" elements.*

Description

Decorates a model object with "preProcess" and "pmml_options" elements.

Usage

```
## Default S3 method:
decorate(x, preProcess = NULL, pmml_options = NULL, ...)
```

Arguments

`x`	The model object.
`preProcess`	A "train::preProcess" object.
`pmml_options`	A list of model type-dependent PMML conversion options.
`...`	Further arguments.

`decorate.earth` *Decorates an "earth" object with an "xlevels" element.*

Description

Decorates an "earth" object with an "xlevels" element.

Usage

```
## S3 method for class 'earth'
decorate(x, data, ...)
```

Arguments

`x`	An "earth" object.
`data`	The training dataset.
`...`	Arguments to pass on to the "decorate.default" function.

decorate.elm *Decorates an "elm" object with a "model" element.*

Description

Decorates an "elm" object with a "model" element.

Usage

```
## S3 method for class 'elm'
decorate(x, data, ...)
```

Arguments

x	An "elm" object.
data	The training dataset.
...	Arguments to pass on to the "decorate.default" function.

decorate.glmnet *Decorates a "glmnet" object with a "lambda.s" element.*

Description

Decorates a "glmnet" object with a "lambda.s" element.

Usage

```
## S3 method for class 'glmnet'
decorate(x, lambda.s, ...)
```

Arguments

x	A "glmnet" object.
lambda.s	The best lambda value. Must be one of listed "glmnet$lambda" values.
...	Arguments to pass on to the "decorate.default" function.

Examples

```
library("glmnet")
library("r2pmml")

data(iris)
iris_X = as.matrix(iris[, -ncol(iris)])
iris_y = iris[, ncol(iris)]
iris.glmnet = glmnet(x = iris_X, y = iris_y, family = "multinomial")
iris.glmnet = decorate(iris.glmnet, lambda.s = iris.glmnet$lambda[49])
r2pmml(iris.glmnet, file.path(tempdir(), "Iris-GLMNet.pmml"))
```

decorate.party *Decorates a "party" object with a "predicted" element.*

Description

Decorates a "party" object with a "predicted" element.

Usage

```
## S3 method for class 'party'
decorate(x, ...)
```

Arguments

x	A "party" object.
...	Arguments to pass on to the "decorate.default" function.

Examples

```
library("evtree")
library("r2pmml")

data(iris)
iris.party = evtree(Species ~ ., data = iris,
    control = evtree.control(max_depth = 3))
iris.party = decorate(iris.party)
r2pmml(iris.party, file.path(tempdir(), "Iris-Party.pmml"))
```

decorate.randomForest *Decorates a "randomForest" object with PMML conversion options.*

Description

Decorates a "randomForest" object with PMML conversion options.

Usage

```
## S3 method for class 'randomForest'
decorate(x, compact = FALSE, ...)
```

Arguments

x	A "randomForest" object.
compact	A flag controlling if decision trees should be transformed from binary splits (FALSE) to multi-way splits (TRUE) representation.
...	Arguments to pass on to the "decorate.default" function.

decorate.ranger *Decorates a "ranger" object with a "variable.levels" element.*

Description

Decorates a "ranger" object with a "variable.levels" element.

Usage

```
## S3 method for class 'ranger'
decorate(x, data, ...)
```

Arguments

x	A "ranger" object.
data	The training dataset.
...	Arguments to pass on to the "decorate.default" function.

Examples

```
library("ranger")
library("r2pmml")

data(iris)
iris.ranger = ranger(Species ~ ., data = iris, num.trees = 17,
    write.forest = TRUE, probability = TRUE)
iris.ranger = decorate(iris.ranger, data = iris)
r2pmml(iris.ranger, file.path(tempdir(), "Iris-Ranger.pmml"))
```

decorate.svm.formula *Decorates a "svm.formula" object with an "xlevels" element.*

Description

Decorates a "svm.formula" object with an "xlevels" element.

Usage

```
## S3 method for class 'svm.formula'
decorate(x, data, ...)
```

Arguments

x	A "svm.formula" object.
data	The training dataset.
...	Arguments to pass on to the "decorate.default" function.

`decorate.train` *Decorates the final model of a "train" object with model type-dependent elements.*

Description

Decorates the final model of a "train" object with model type-dependent elements.

Usage

```
## S3 method for class 'train'
decorate(x, ...)
```

Arguments

`x`	A "train" object.
`...`	Arguments to pass on to the "decorate.default" function.

`decorate.WrappedModel` *Decorates a "WrappedModel" object with "invert_levels" element. Additionally, decorates the learned model with model type-dependent elements.*

Description

Decorates a "WrappedModel" object with "invert_levels" element. Additionally, decorates the learned model with model type-dependent elements.

Usage

```
## S3 method for class 'WrappedModel'
decorate(x, invert_levels = FALSE, ...)
```

Arguments

`x`	A "WrappedModel" object.
`invert_levels`	A flag indicating if the learned model should assume normal (FALSE) or inverted (TRUE) ordering of category values for the binary categorical target field.
`...`	Arguments to pass on to the "decorate.default" function

`decorate.xgb.Booster` *Decorates an "xgb.Booster" object with "fmap", "schema", "ntreelimit" and "pmml_options" elements.*

Description

Decorates an "xgb.Booster" object with "fmap", "schema", "ntreelimit" and "pmml_options" elements.

Usage

```
## S3 method for class 'xgb.Booster'
decorate(
  x,
  fmap,
  response_name = NULL,
  response_levels = c(),
  missing = NULL,
  ntreelimit = NULL,
  compact = FALSE,
  ...
)
```

Arguments

`x`	An "xgb.Booster" object.
`fmap`	An XGBoost feature map as a "data.frame" object.
`response_name`	The name of the target field.
`response_levels`	A list of category values for a categorical target field.
`missing`	The string representation of missing input field values.
`ntreelimit`	The number of decision trees (aka boosting rounds) to convert.
`compact`	A flag controlling if decision trees should be transformed from binary splits (FALSE) to multi-way splits (TRUE) representation.
`...`	Arguments to pass on to the "decorate.default" function.

Examples

```
library("xgboost")
library("r2pmml")

data(iris)
iris_X = iris[, -ncol(iris)]
iris_y = iris[, ncol(iris)]
# Convert from factor to integer[0, num_class]
iris_y = (as.integer(iris_y) - 1)
```

```
iris.matrix = model.matrix(~ . - 1, data = iris_X)
iris.DMatrix = xgb.DMatrix(iris.matrix, label = iris_y)
iris.fmap = as.fmap(iris.matrix)
iris.xgboost = xgboost(data = iris.DMatrix,
    objective = "multi:softprob", num_class = 3, nrounds = 11)
iris.xgboost = decorate(iris.xgboost, iris.fmap,
    response_name = "Species", response_levels = c("setosa", "versicolor", "virginica"))
pmmlFile = file.path(tempdir(), "Iris-XGBoost.pmml")
r2pmml(iris.xgboost, pmmlFile, compact = FALSE)
compactPmmlFile = file.path(tempdir(), "Iris-XGBoost-compact.pmml")
r2pmml(iris.xgboost, compactPmmlFile, compact = TRUE)
```

`r2pmml` *Converts an R model object to PMML.*

Description

Converts an R model object to PMML.

Usage

```
r2pmml(
  x,
  file,
  schema = NULL,
  converter = NULL,
  converter_classpath = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	An R model object.
`file`	A filesystem path to the result file.
`schema`	The PMML schema version for the PMML document.
`converter`	The name of a custom JPMML-R converter class.
`converter_classpath`	A list of filesystem paths to library JAR files that provide and support the custom JPMML-R converter class.
`verbose`	A flag controlling the verbosity of the conversion process.
`...`	Arguments to be passed on to the "r2pmml::decorate" function.

Examples

```
library("mlbench")
library("randomForest")
library("r2pmml")

data(iris)
iris.rf = randomForest(Species ~ ., data = iris, ntree = 7)
# Convert "randomForest" object to R-style (deep binary splits) MiningModel
pmmlFile = file.path(tempdir(), "Iris-RandomForest.pmml")
r2pmml(iris.rf, pmmlFile)
# Convert "randomForest" object to PMML-style (shallow multi-way splits) MiningModel
compactPmmlFile = file.path(tempdir(), "Iris-RandomForest-compact.pmml")
r2pmml(iris.rf, compactPmmlFile, compact = TRUE)

data(BostonHousing)
housing.glm = glm(medv ~ ., data = BostonHousing, family = "gaussian")
# Convert "glm" object into GeneralRegressionModel
genRegPmmlFile = file.path(tempdir(), "Housing-GLM.pmml")
r2pmml(housing.glm, genRegPmmlFile)
# Convert "glm" object into RegressionModel
regPmmlFile = file.path(tempdir(), "Housing-LM.pmml")
r2pmml(housing.glm, regPmmlFile, converter = "org.jpmml.rexp.LMConverter")
```

---

verify *Dispatches execution to the most appropriate model verification function.*

Description

Dispatches execution to the most appropriate model verification function.

Usage

```
verify(x, newdata, ...)
```

Arguments

x	A model object.
newdata	The verification dataset.
...	Arguments to pass on to the selected function.

verify.default *Enhances a model object with verification data.*

Description

Enhances a model object with verification data.

Usage

```
## Default S3 method:
verify(x, newdata, ...)
```

Arguments

x	A model object.
newdata	The verification dataset.
...	Further arguments.

verify.glm *Enhances a "glm" object with verification data.*

Description

Enhances a "glm" object with verification data.

Usage

```
## S3 method for class 'glm'
verify(x, newdata, precision = 1e-13, zeroThreshold = 1e-13, ...)
```

Arguments

x	A "glm" object.
newdata	The verification dataset.
precision	Maximal relative error.
zeroThreshold	Maximal absolute error near the zero value.
...	Further arguments.

Examples

```
library("mlbench")
library("r2pmml")

data(BostonHousing)
housing.glm = glm(medv ~ ., data = BostonHousing, family = "gaussian")
housing.glm = verify(housing.glm, newdata = BostonHousing[sample(nrow(BostonHousing), 10), ])
r2pmml(housing.glm, file.path(tempdir(), "Housing-GLM-verified.pmml"))
```

---

verify.train *Enhances a "train" object with verification data.*

Description

Enhances a "train" object with verification data.

Usage

```
## S3 method for class 'train'
verify(x, newdata, precision = 1e-13, zeroThreshold = 1e-13, ...)
```

Arguments

x	A "train" object.
newdata	The verification dataset.
precision	Maximal relative error.
zeroThreshold	Maximal absolute error near the zero value.
...	Arguments to pass on to the "predict.train" method.

---

verify.xgb.Booster *Enhances an "xgb.Booster" object with verification data.*

Description

Enhances an "xgb.Booster" object with verification data.

Usage

```
## S3 method for class 'xgb.Booster'
verify(
  x,
  newdata,
  precision = 1e-06,
  zeroThreshold = 1e-06,
  response_name = NULL,
  response_levels = c(),
  ...
)
```

Arguments

x	An "xgb.Booster" object.
newdata	The verification dataset.
precision	Maximal relative error.
zeroThreshold	Maximal absolute error near the zero value.
response_name	The name of the target field.
response_levels	A list of category values for a categorical target field.
...	Arguments to pass on to the "predict.xgb.Booster" method.

---

write.fmap *Writes XGBoost feature map to a file.*

Description

Writes XGBoost feature map to a file.

Usage

```
write.fmap(fmap, file)
```

Arguments

fmap	An XGBoost feature map as a "data.frame" object.
file	A filesystem path to the result file.

Index