Title  Create the Best Train for Classification Models

Version  0.1.5

Description  Patterns searching and binary classification in economic and financial data is a large field of research. There are a large part of the data that the target variable is binary. Nowadays, many methodologies are used, this package collects most popular and compare different configuration options for Linear Models (LM), Generalized Linear Models (GLM), Linear Mixed Models (LMM), Discriminant Analysis (DA), Classification And Regression Trees (CART), Neural Networks (NN) and Support Vector Machines (SVM).

Depends  R (>= 3.2.3)

License  GPL (>= 2)

BugReports  https://github.com/economistgame/OptimClassifier/issues

URL  https://economistgame.github.io/OptimClassifier

Imports  crayon, dplyr, MASS, lme4, rpart, nnet, e1071, lmtest, nortest, clisymbols, ggplot2

Encoding  UTF-8

LazyData  true

RoxygenNote  6.1.1

Suggests  testthat, knitr, rmarkdown

VignetteBuilder  knitr

NeedsCompilation  no

Author  Agustin Perez-Martín [aut] (<https://orcid.org/0000-0003-4994-3176>), Agustin Perez-Torregrosa [cre, aut] (<https://orcid.org/0000-0001-5658-4795>), Marta Vaca-Lamata [aut] (<https://orcid.org/0000-0001-8496-5579>), Antonio Jose Verdu-Jover [aut] (<https://orcid.org/0000-0002-6201-7196>)

Maintainer  Agustin Perez-Torregrosa <agustin.perez@oumh.umh.es>

Repository  CRAN

Date/Publication  2020-01-14 09:20:10 UTC
**R topics documented:**

- AustralianCredit .......................................................... 2
- MC ............................................................................. 3
- Microsoft ................................................................. 5
- Optim.CART ............................................................ 6
- Optim.DA ................................................................. 7
- Optim.GLM .............................................................. 8
- Optim.LM ................................................................. 9
- Optim.LMM ............................................................ 10
- Optim.NN ................................................................. 11
- Optim.object ........................................................... 12
- Optim.SVM ............................................................. 13
- print.Optim ............................................................. 14
- RMSE ......................................................................... 15
- sampler .................................................................... 15

**Index**

<table>
<thead>
<tr>
<th>AustralianCredit</th>
<th>A Credit Approval Dataset</th>
</tr>
</thead>
</table>

**Description**

This dataset concerns credit card applications and represent positive and negative instances of people who were and were not granted credit. It has served as an important test data set for several credit-scoring algorithms. This dataset was introduced by Quinlan (1987).

**Usage**

data("AustralianCredit")

**Format**

A data frame with 690 observations on the following 15 variables.

- X1 a factor with levels 0 and 1
- X2 a numeric vector
- X3 a numeric vector
- X4 a factor with 3 levels
- X5 a factor with 14 levels
- X6 a factor with 9 levels
- X7 a numeric vector
- X8 a factor with levels 0 and 1
- X9 a factor with levels 0 and 1
- X10 a numeric vector
X11 a factor with levels 0 and 1
X12 a factor with 3 levels
X13 a numeric vector
X14 a numeric vector
Y a factor with levels 0 and 1

References

Examples

data(AustralianCredit)

## See a general view of a dataset
summary(AustralianCredit)

## Plot a response variable
plot(AustralianCredit$Y)

---

MC

Confusion Matrix

Description
Confusion Matrix is a contingency table that gives a visualization of the performance of an algo-
rithm

Usage
MC(yhat, y, metrics = FALSE)

Arguments
yhat A predicted value vector.
y A real value vector.
metrics Calculate all metrics. See details for more information.

Details
Also it known as an error matrix. Normally, you can identify 4 elements, they known as true
positive (TP), true negative (TN), false positive (FP) and false negative (FN). To understand it, a
simple example is presented:

<table>
<thead>
<tr>
<th>Real Values</th>
<th>Estimated</th>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The problem arises that there is not always a clear relationship between which is the positive class or there may be different classes so it is also common to use the terms Type I error (FP), Type II error (FN) and unify the success or accuracy (TP+TN) in a single value.

Suppose a 3x3 table with notation

<table>
<thead>
<tr>
<th></th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Class 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class 2</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>Class 3</td>
<td>G</td>
<td>H</td>
<td>I</td>
</tr>
</tbody>
</table>

where \( N = A+B+C+D+E+F+G+H+I \)

The formulas used here are:

- **Successrate** = \( (A + E + I)/N \)
- **TypeIerror** = \( (B + F + C)/N \)
- **TypeIIerror** = \( (D + H + G)/N \)

Other indicators depends of one class and in the case choose Class 1

- **SensitivityClass1** = \( A/(A + D + G) \)
- **SpecificityClass1** = \( (E + I)/(B + E + H + C + F + I) \)
- **PrecisionClass1** = \( A/(A + E + I) \), also it is called Positive Predictive Value (PPV)
- **PrevalenceClass1** = \( (A + D + G)/N \)

**References**


**Examples**

```r
if(interactive()){
  # You can create a confusion Matrix like a table
  RealValue <- c(1,0,1,0)
  Predicted <- c(1,1,1,0)
  MC(y = RealValue, yhat=Predicted ,metrics=TRUE)
}
```
Microsoft

Daily Information Data of Microsoft

Description
Daily Information Data of Microsoft, 2007/01/03~2018-03-13

Usage
data("Microsoft")

Format
A data frame with 2817 observations on the following 6 variables.

Date  a Date
Y    a factor with levels 0 and 1
DayWeek  a factor, represent the day of the week
Month  a factor, month
LastDay  a numeric vector, difference of open price and close price
PayDiv  a logical vector, represent when Microsoft was payed dividends

Source
Yahoo Finance.

Examples
data(Microsoft)

## See a general view of a dataset
summary(Microsoft)

## Plot a response variable
plot(Microsoft$Y)
Tune CART for the optimal complexity parameter

Description

The complexity parameter aims to save computing time by pruning off splits that are obviously not worthwhile. This function starting with null value of cp and ranks the different possibly levels of pruning trees find best CART for different levels of cost complexity. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile.

Usage

Optim.CART(formula, data, p, includedata = FALSE, seed = NULL, ...)

Arguments

- **formula**: A formula of the form \( y \sim x_1 + x_2 + \ldots \)
- **data**: Data frame from which variables specified in formula are preferentially to be taken.
- **p**: A percentage of training elements
- **includedata**: Logicals. If TRUE the training and testing datasets are returned.
- **seed**: A single value, interpreted as an integer, or NULL. The default value is NULL, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.
- **...**: Arguments passed to \texttt{rpart}

Details

Classification And Regression Tree (CART) are a decision tree learning technique that produces either classification or regression trees, first introduced by Breiman et al. (1984). Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.

Value

An object of class \texttt{Optim}. See \texttt{Optim.object}

References

Examples
if(interactive()){
  ## Load a Dataset
  data(AustralianCredit)
  ## Generate a model
  modelFit <- Optim.CART(Y~., AustralianCredit, p = 0.7, seed=2018)
  modelFit
}

Optim.DA

Discover the best Discriminant Analysis for your data

Description
This function search the best Discriminant Analysis (DA) between LDA and QDA.

Usage
Optim.DA(formula, data, p, criteria = c("rmse", "success_rate", "ti_error", "tii_error"), includedata = FALSE, seed = NULL, ...)

Arguments

  formula A formula of the form y ~ x1 + x2 + ...

  data Data frame from which variables specified in formula are preferentially to be taken.

  p A percentage of training elements

  criteria Select criterion to use.

  includedata logicals. If TRUE the training and testing datasets are returned.

  seed a single value, interpreted as an integer, or NULL. The default value is NULL, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.

  ... arguments passed to lda and qda

Details
LDA and QDA are distribution-based classifiers with the assumption that data follows a multivariate normal distribution. LDA differs from QDA in the assumption about the class variability. LDA assumes that all classes share the same within-class covariance matrix whereas QDA allows for distinct within-class covariance matrices.

Value
An object of class Optim. See Optim.object
Examples

```r
if(interactive()){
  ## Load a Dataset
  data(AustralianCredit)
  ## Generate a Model
  modelFit <- Optim.DA(Y~., AustralianCredit, p = 0.7, seed=2018)
  modelFit
}
```

**Optim.GLM**

*Find out what is the error distribution and link function that best fits a classification generalized linear model to your data*

**Description**

*Optim.GLM* is used to fit the best classification GLM to a dataset. For this purpose, we examine the variation of the precision using the root mean square error (RMSE) when different error distribution and link function was used in the model. In addition, several thresholds are applied to check which is the most optimal cut for the indicators derived from the confusion matrix (success rate, type I error and type II error) according to a given criterion.

**Usage**

```r
Optim.GLM(formula, data, p, criteria = c("success_rate", "ti_error", "tii_error"), includedata = FALSE, seed = NULL, ...)
```

**Arguments**

- **formula**: A formula of the form `y ~ x1 + x2 + ...`
- **data**: Data frame from which variables specified in formula are preferentially to be taken.
- **p**: A percentage of training elements
- **criteria**: This variable selects the criteria to select the best threshold. The default value is `success_rate`
- **includedata**: logicals. If TRUE the training and testing datasets are returned.
- **seed**: a single value, interpreted as an integer, or NULL. The default value is NULL, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.
- **...**: arguments passed to `glm`

**Value**

An object of class Optim. See *Optim.object*
Examples

```r
if(interactive()){
  ## Load a Dataset
  data(AustralianCredit)

  ## Create the model
  creditscoring <- Optim.GLM(Y~., AustralianCredit, p = 0.7, seed=2018)

  #See a ranking of the models tested
  print(creditscoring)

  #Access to summary of the best model
  summary(creditscoring)

  #not sure of like the best model, you can access to the all model, for example the 2nd model
  summary(creditscoring,2)
}
```

---

### Optim.LM

*Find out what is the transformation of the response variable that best fits a classification linear model to your data*

#### Description

`Optim.LM` is used to fit the best classification linear model to a dataset. For this purpose, we examine the variation of the precision using the root mean square error (RMSE) when transformations are applied on the response variable. In addition, several thresholds are applied to check which is the most optimal cut for the indicators derived from the confusion matrix (success rate, type I error and type II error) according to a given criterion.

#### Usage

```r
Optim.LM(formula, data, p, seqthreshold = 0.05, criteria = c("success_rate", "error_ti", "error_tii"), includedata = FALSE, seed = NULL, ...)
```

#### Arguments

- **formula**: A formula of the form `y ~ x1 + x2 + ...`
- **data**: Data frame from which variables specified in `formula` are preferentially to be taken.
- **p**: A percentage of training elements
- **seqthreshold**: Linear models doesn’t return a class, it returns probability because of he must cut by levels. This parameter allows you to select the percentage between one threshold and next evaluated.
criteria

This variable selects the criteria to select the best threshold. The default value is
success_rate

includedata

logicals. If TRUE the training and testing datasets are returned.

seed

a single value, interpreted as an integer, or NULL. The default value is NULL, but
for future checks of the model or models generated it is advisable to set a random
seed to be able to reproduce it.

... arguments passed to lm

Value

An object of class Optim. See Optim.object

Examples

if(interactive()){
  ## Load a Dataset
  data(AustralianCredit)

  ## Create the model
  linearcreditscoring <- Optim.LM(Y~., AustralianCredit, p = 0.7, seed=2018)

  #See a ranking of the models tested
  print(linearcreditscoring)

  #Access to summary of the best model
  summary(linearcreditscoring)

  #not sure of like the best model, you can access to the all model, for example the 2nd model
  summary(linearcreditscoring,2)
}

---

Optim.LMM

Discover what is the best random variable for your data set

Description

This function allows to find best LMM for a specific data.

Usage

Optim.LMM(response, data, p, criteria = c("success_rate", "error_ti", "error_tii"), randomatributtecandidate = NULL, includedata = FALSE, seed = NULL, ...)
Optim.NN

**Arguments**

- **response**: A character object that contains the name of the response variable about which a researcher is asking a question. "Y"
- **data**: Data frame from which variables specified in formula are preferentially to be taken.
- **p**: A percentage of training elements
- **criteria**: This variable selects the criteria to select the best threshold. The default value is `success_rate`
- **randomattributecandidate**: a character vector, or `NULL`. The default value is `NULL`, the function tests with all those categorical variables in the data. The default option is not recommended. Because the decision must be made according to the objective of statistical modeling. But it can serve as orientation.
- **includedata**: logicals. If TRUE the training and testing datasets are returned.
- **seed**: a single value, interpreted as an integer, or `NULL`. The default value is `NULL`, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.
- **...**: arguments passed to `lmer`

**Value**

An object of class Optim. See `Optim.object`

**Examples**

```r
if(interactive()){
  ## Load a Dataset
  data(AustralianCredit)
  ## Generate a model
  modelFit <- Optim.LMM("Y", AustralianCredit, p = 0.7, seed=2018)
  modelFit
}
```

---

**Description**

Optim.NN function allows finding the best NN.

**Usage**

```r
Optim.NN(formula, data, p, criteria = c("success_rate", "ti_error", "tii_error"), includedata = FALSE, seed = NULL, maxhiddenlayers = 10, maxit = 500, MaxNWts = 2000, ...)
```
Arguments

- **formula**: A formula of the form $y \sim x_1 + x_2 + \ldots$
- **data**: data frame from which variables specified in formula are preferentially to be taken.
- **p**: a percentage of training elements
- **criteria**: this variable selects the criteria to select the best threshold. The default value is success_rate
- **includedata**: logicals. If TRUE the training and testing datasets are returned.
- **seed**: a single value, interpreted as an integer, or NULL. The default value is NULL, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.
- **maxhiddenlayers**: the high number of hidden layers for the neural network considers.
- **maxit**: the maximum allowable number of weights. There is no intrinsic limit in the code, but increasing MaxNWts will probably allow fits that are very slow and time-consuming.
- **MaxNWts**: maximum number of iterations. Default 500.
- **...**: arguments passed to nnet

Value

An object of class Optim. See Optim.object

Examples

```r
if(interactive()){
  ## Load a Dataset
  data(AustralianCredit)
  ## Generate a model
  modelFit <- Optim.NN(Y~., AustralianCredit, p = 0.7, seed=2018)
  modelFit
}
```

---

**Optim.object**

*Optimized Classifier Object*

Description

These are objects representing different fitted models.
Value

Type character string: the method used to fit the model. At the moment the following models are implemented: "LM" (lm), "GLM" (glm), "LMM" (lmer), "CART" (rpart), "DA" (lda and qda), "NN" (nnet) and "SVM" (svm).

Models a data.frame whose content summarize the different models generated, ordered for selected criterion

Model a list of the models generated

Predict a list of the predicts generated

Thresholds a list whose content data.frames that summarize the different thresholds tested. This component is only available in LM, GLM, NN and SVM

Confussion_Matrices a data.frame whose content summarize the different models generated

Data a list which training and testing datasets

Inference Tests a data.frame with different diagnostics for models generated. It is only available in LM

Structure

The following components must be included in a legitimate Optim object.

Optim.SVM

Discover the best SVM for your data

Description

This function allows to find the best kernel for tune your support vector machine (SVM).

Usage

Optim.SVM(formula, data, p, criteria = c("rmse", "success", "ti_error", "tii_error"), includedata = FALSE, seed = NULL, ...)

Arguments

formula A formula of the form y ~ x1 + x2 + ...

data Data frame from which variables specified in formula are preferentially to be taken.

p A percentage of training elements

criteria This variable selects the criteria to select the best threshold. The default value is success_rate.

includedata logicals. If TRUE the training and testing datasets are returned.

seed a single value, interpreted as an integer, or NULL. The default value is NULL, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.

... arguments passed to svm
Value

An object of class Optim. See Optim.object

Examples

if(interactive()){

## Load a Dataset
data(AustralianCredit)

## Generate a model
modelFit <- Optim.SVM(Y~., AustralianCredit, p = 0.7, seed=2018)
modelFit

}

print.Optim

Print an Optim Object

Description

This function prints an Optim object. It is a method for the generic function print of class "Optim".

Usage

## S3 method for class 'Optim'
print(x, plain = FALSE, digits = getOption("digits"),
     ...)

Arguments

x           object of class "Optim"
plain       select if you want enriched output mode (with colors and bold) or a plain output mode.
digits      minimal number of significant digits.
...          further arguments passed to or from other methods.
RMSE

Root Mean Square Error

**Description**

RMSE is a commonly used error metric to measure the performance of regression models, but it is also possible to use it in a classification system. The RMSE measures the standard deviation of the predictions from the ground-truth. This is the relationship between RMSE and classification.

**Usage**

```r
RMSE(yhat, y, type.of = c("numeric", "text", "scalable"))
```

**Arguments**

- `yhat` A predicted value vector
- `y` A real value vector
- `type.of` Type of response variable, either: numeric for the numerical response variables, text for the class response variables without growing relationship or scalable for the class response variables without growing relationship.

---

**sampler**

*Splitting your dataset in training and testing*

**Description**

A training/test partition are created by `sampler` function.

**Usage**

```r
sampler(data, p, seed = NULL)
```

**Arguments**

- `data` Data frame from which all variables
- `p` The percentage of data that goes to training. It can be expressed in either decimal fraction (such as 0.7) or percent (such as 72.12).
- `seed` a single value, interpreted as an integer, or `NULL`. The default value is `NULL`, but for future checks of the model or models generated it is advisable to set a random seed to be able to reproduce it.
Examples

if(interactive()){
  # The best way to demonstrate the functionality is test the function

  Sampling <- sampler(AustralianCredit,p=0.7)
}

Index

*Topic datasets
  AustralianCredit, 2
  Microsoft, 5

*Topic methods
  Optim.object, 12

AustralianCredit, 2

glm, 8, 13

lda, 7, 13
lm, 10, 13
lmer, 11, 13

MC, 3
Microsoft, 5

nnet, 12, 13

Optim.CART, 6
Optim.DA, 7
Optim.GLM, 8
Optim.LM, 9
Optim.LMM, 10
Optim.NN, 11
Optim.object, 6–8, 10–12, 12, 14
Optim.SVM, 13

print.Optim, 14

qda, 7, 13

RMSE, 15
rpart, 6, 13

sampler, 15
svm, 13