Package ‘gencve’

April 12, 2016

Type Package
Title General Cross Validation Engine
Version 0.3
Author A. I. McLeod
Maintainer A. I. McLeod <aimcleod@uwo.ca>
Description Engines for cross-validation of many types of regression and class prediction models are provided. These engines include built-in support for 'glmnet', 'lars', 'plus', 'MASS', 'rpart', 'C50' and 'randomforest'. It is easy for the user to add other regression or classification algorithms. The 'parallel' package is used to improve speed. Several data generation algorithms for problems in regression and classification are provided.
License GPL (>= 2)
LazyData TRUE
Depends R (>= 2.10)
Imports lars, glmnet, plus, C50, randomForest, e1071, rpart, MASS, class, nnet
URL http://www.stats.uwo.ca/faculty/aim
NeedsCompilation no
Repository CRAN
Date/Publication 2016-04-12 00:56:56

R topics documented:

gencve-package .................................................. 2
cgcv ................................................................. 5
churn ................................................................. 6
Detroit ................................................................. 7
dShao ................................................................. 9
featureSelect ....................................................... 10
fires ................................................................. 11
gcv ................................................................. 12
kNN_LOOCV ........................................................ 13
kNN_MLE ..................................................... 14
kyphosis ............................................... 16
logloss ................................................... 16
mae ....................................................... 17
mape ..................................................... 18
meatspec ................................................. 19
misclassificationrate ................................... 19
mse ......................................................... 20
pollution ............................................... 21
prostate ............................................... 22
rdigitsBFOS ............................................. 23
regal ..................................................... 24
rmix ....................................................... 25
xor ........................................................ 26
ShaoReg ............................................... 27
SinghTest .............................................. 28
SinghTrain ............................................. 32
smape .................................................... 35
vifx ....................................................... 36
yhat_CART .............................................. 37
yhat_gel ............................................... 38
yhat_lars .............................................. 39
yhat_lm .................................................. 40
yhat_nn .................................................. 41
yhat_plus .............................................. 42
yhat_RF ............................................... 43
yhat_step .............................................. 44
yhat_SVM .............................................. 45
yh_C50 .................................................. 46
yh_CART ............................................... 46
yh_kNN .................................................. 47
yh_lda .................................................. 48
yh_logistic ............................................ 49
yh_NB ................................................... 50
yh_qda .................................................. 50
yh_RF ................................................... 51
yh_svm .................................................. 52

Index 53
Description

Engines for cross-validation of many types of regression and class prediction models are provided. These engines include built-in support for 'glmnet', 'lars', 'plus', 'MASS', 'rpart', 'C50' and 'randomforest'. It is easy for the user to add other regression or classification algorithms. The 'parallel' package is used to improve speed. Several data generation algorithms for problems in regression and classification are provided.

Details

The DESCRIPTION file:

Index of help topics:

Detroit
ShaoReg
SinghTest
SinghTrain
cgcv
churnTrain
dShao
featureSelect
fires
gcv
genCV
kNN_LOOCV
kNN_MLE
kyphosis
logloss
mae
mape
meatspec
misclassification_rate
mse
pollution
prostate
rdigitsBFOS
regal
rmix
rxor
smape
vifx
yh_C50
yh_CART
yh_NB
yh_RF

Detroit Homicide Data for 1961–73
Synthetic Regression Data
Singh Prostate Microarray Test Data
Singh Prostate Microarray Training Data
Estimate Misclassification Rate Using d-fold Cross-Validation for Class Prediction
Customer Churn Data
Shao Holdout Sample Size for Linear Regression Variable Selection
Feature Select For Wide Data
Forest Fires in Montesinho Natural Park
Estimate EPE Using Delete-d Cross-Validation
General Cross Validation Engine General Cross Validation Engine
Select k with Leave-one-out CV
MLE k in kNN
Data on Children who have had Corrective Spinal Surgery
log-loss function for multiclass prediction
Mean Absolute Error
Mean Absolute Percentage Error
Meat Spectrometry to Determine Fat Content
Misclassification Rate for Class Prediction
Mean Square Error Loss
Pollution Data from McDonald and Schwing
Prostate Cancer Data
BFOS Digit Recognition Problem
Regression EPE for All Implemented Methods
Random Mixture Classification Example
Random XOR Samples
Mean Absolute Percentage Error
Variance Inflation Factor
C50 Prediction
CART Prediction
Naive Bayes Prediction
Random Forest Prediction
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>yh_kNN</code></td>
<td>kNN or NN prediction</td>
</tr>
<tr>
<td><code>yh_lda</code></td>
<td>LDA predictions</td>
</tr>
<tr>
<td><code>yh_logistic</code></td>
<td>Logistic Regression and Regularized Logistic Regression Prediction</td>
</tr>
<tr>
<td><code>yh_qda</code></td>
<td>QDA Prediction</td>
</tr>
<tr>
<td><code>yh_svm</code></td>
<td>Support Vector Machine Prediction</td>
</tr>
<tr>
<td><code>yhat_CART</code></td>
<td>CART regression prediction</td>
</tr>
<tr>
<td><code>yhat_RF</code></td>
<td>Fit Random Forest Regression Predictor</td>
</tr>
<tr>
<td><code>yhat_SVM</code></td>
<td>Support Vector Machine Regression Prediction</td>
</tr>
<tr>
<td><code>yhat_gel</code></td>
<td>Elastic Net Regression Prediction</td>
</tr>
<tr>
<td><code>yhat_lars</code></td>
<td>Fit LASSO Regression using Mallows Cp and Predict</td>
</tr>
<tr>
<td><code>yhat_lm</code></td>
<td>Linear Predictor using Least-Squares Regression</td>
</tr>
<tr>
<td><code>yhat_nn</code></td>
<td>Nearest Neighbour Prediction</td>
</tr>
<tr>
<td><code>yhat_plus</code></td>
<td>SCAD or MCP Regression Prediction</td>
</tr>
<tr>
<td><code>yhat_step</code></td>
<td>Backward Stagewise Regression with AIC or BIC</td>
</tr>
</tbody>
</table>

Engines for cross-validation of many types of regression and class prediction models are provided. These engines include built-in support for CRAN packages including glmnet, lars, plus, MASS, rpart, C50 and randomforest. The cross validation engines are the functions `gcv()` and `cgcv()`. It is easy for the user to add other regression or classification algorithms for use with these engines. The default cost function for regression is squared error but support is provided for mean absolute error and mean percentage absolute error. For classification the default cost function 0/1 loss with the associated mis-classification rate but logloss is also provided. The user may also specify their own cost function. Both `gcv()` and `cgcv()` make use of R’s parallel package. Several illustrative datasets are included as well as data generation algorithms for problems in regression and classification.

The delete-d cross validation method of Shao (1993) is used. Shao recommends at least 1000 iterations so this method requires significantly more computation than k-fold cross-validation that is recommend by Hastie, Tibshirani and Friedman (2009), in conjunction with regularization using the one-standard-deviation rule, for the purpose of selecting a tuning parameter in penalized regression. However many researchers have noticed that even regularized k-fold cross-validation is quite variable (Kim, 2009). A future version of this package will include k-fold cross-validation and iterated k-fold cross-validation. Usually iterated k-fold cross-validation produces very similar results to the delete-d method (Kim, 2009).

Other CRAN packages that provide general frameworks with resampling strategies include boot, mlr and caret.

**Author(s)**

A. I. McLeod
Maintainer: A. I. McLeod <aimcleod@uwo.ca>

**References**

Trevor Hastie, Robert Tibshirani, Jerome H. Friedman (2009), The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Ed. Springer.


See Also
cv.glm

Examples

#Regression with simulated model
xy <- ShaoReg()
gcv(xy[,1:8], xy[,9], MaxIter=25, d=5)
#
#SVM with simulated mixture data
xy <- rmix(100)
cgcv(X=xy[,1:2], y=xy[,3], yh=yh_svm, MaxIter=25)
#
#data has been divided into training and test just do simple
# cross-validation
yh_CART(SinghTrain, SinghTest)

cgcv

Estimate Misclassification Rate Using d-fold Cross-Validation for Class Prediction

Description

This is a general purpose function to estimate the misclassification rate for a specified classifier.

Usage
cgcv(X, y, yh = yh_NN, MaxIter = 1000, d = ceiling(length(y)/10), NCores = 1, libs = character(0), seed = "default", ...)

Arguments

X inputs
y output factor
yh function with arguments dfTrain and dfTest that produces the misclassification rate for test data
MaxIter Number of iterations of the CV procedure
d Number of observations for the hold-out sample
NCores Default is 1 which does not use the parallel package. Otherwise, you can set to the number of cores available. If unsure, just experiment!
libs Required libraries needed for the predictor.
seed Default is to use R’s default which is based on the current time. Otherwise set to an integer value. See Details.
... Additional arguments that are passed to yh.
Value

cross-validated mis-classification rate

Author(s)

A. I. McLeod

Examples

\[
X_y \leftarrow \text{rmix}(200) \ # \text{training data} \\
X \leftarrow \text{as.matrix.data.frame}(X_y[,1:2]) \\
y \leftarrow X_y[,3] \\
cgcv(X, y, \text{MaxIter}=50)
\]

---

**Churn**

Customer Churn Data

Description

A data set from the MLC++ machine learning software for modeling customer churn. There are 19 predictors, mostly numeric: state (categorical), account_length, area_code, international_plan (yes/no), voice_mail_plan (yes/no), number_vmail_messages, total_day_minutes, total_day_calls, total_day_charge, total_eve_minutes, total_eve_calls, total_eve_charge, total_night_minutes, total_night_calls, total_night_charge, total_intl_minutes, total_intl_calls, total_intl_charge and number_customer_service_calls.

The outcome is contained in a column called churn (also yes/no).

The training data has 3333 samples and the test set contains 1667.

A note in one of the source files states that the data are "artificial based on claims similar to real world".

A rule-based model shown on the RuleQuest website contains 19 rules, including:

**Rule 1:** (2221/60, lift 1.1)

\[ \text{international plan} = \text{no} \]
\[ \text{total day minutes} \leq 223.2 \]
\[ \text{number customer service calls} \leq 3 \]
\[-\rightarrow \text{class 0 [0.973]}\]

**Rule 5:** (1972/87, lift 1.1)

\[ \text{total day minutes} \leq 264.4 \]
\[ \text{total intl minutes} \leq 13.1 \]
\[ \text{total intl calls} > 2 \]
\[ \text{number customer service calls} \leq 3 \]
\[-\rightarrow \text{class 0 [0.955]}\]

**Rule 10:** (60, lift 6.8)

\[ \text{international plan} = \text{yes}\]
Detroit Homicide Data for 1961-73

For convenience we have labelled the input variables 1 through 11 to be consistent with the notation used in Miller (2002). Only the first 11 variables were used in Miller’s analyses. The best fitting subset regression with these 11 variables, uses only 3 inputs and has a residual sum of squares of 6.77 while using forward selection produces a best fit with 3 inputs with residual sum of squares 21.19. Backward selection and stagewise methods produce similar results. It is remarkable that there is such a big difference. Note that the usual forward and backward selection algorithms may fail since the linear regression using 11 variables gives essentially a perfect fit.
**Format**

A data frame with 13 observations on the following 14 variables.

- FTP. 1 Full-time police per 100,000 population
- UEMP. 2 Percent unemployed in the population
- MAN. 3 Number of manufacturing workers in thousands
- LIC. 4 Number of handgun licences per 100,000 population
- GR. 5 Number of handgun registrations per 100,000 population
- CLEAR. 6 Percent homicides cleared by arrests
- WM. 7 Number of white males in the population
- NMAN. 8 Number of non-manufacturing workers in thousands
- GOV. 9 Number of government workers in thousands
- HE. 10 Average hourly earnings
- WE. 11 Average weekly earnings
- ACC. Death rate in accidents per 100,000 population
- ASR. Number of assaults per 100,000 population
- HOM. Number of homicides per 100,000 of population

**Details**

The data were originally collected and discussed by Fisher (1976) but the complete dataset first appeared in Gunst and Mason (1980, Appendix A). Miller (2002) discusses this dataset throughout his book. The data were obtained from the StatLib data archive.

**References**


**Examples**

```r
#Detroit data example
data(Detroit)
str(Detroit)
```
Description

Implements Shao’s recommendation.

Usage

dShao(n)

Arguments

n number of observations

Value

hold-out sample size for delete-d method. This is the validation data

Note

This is only recommended in the variable selection problem in the classical asymptotic linear regression setting where p is fixed and n is much larger than p, where n is the number of observations and p is the number of independent variables.

Author(s)

A. I. McLeod

References


See Also

gcv

Examples

dShao(100)
featureSelect  

Feature Select For Wide Data  

Description  

A commonly used method with microarrays to select the best genes for class prediction is implemented. This method involves computing the one-way anova for each gene and select the genes with the between classes sum of squares or equivalently the largest F-ratios.

Usage  

featureSelect(X, y, numFeatures = 10)

Arguments  

- **X**: data matrix  
- **y**: must be a factor with length equal to the number of rows of X  
- **numFeatures**: the number of features to be selected - usually larger than the default 10.

Value  

the column indices corresponding to the columns of X that are selected

Author(s)  

A. I. McLeod

References  

tba

Examples  

Xy <- churnTrain  
y <- Xy[, ncol(Xy)]  
Xy <- Xy[, -ncol(Xy)]  
X <- as.matrix.data.frame(Xy[, -(1:5)])  
(ind <- featureSelect(X, y, numFeatures=5))  
colnames(X)[ind]
Forest Fires in Montesinho Natural Park

**Description**

The forest fire data were collected during January 2000 to December 2003 for fires in the Montesinho natural park located in the northeast region of Portugal. The response variable of interest was area burned in ha. When the area burned as less than one-tenth of a hectare, the response variable was set to zero. In all there were 517 fires and 247 of them recorded as zero. The region was divided into a 10-by-10 grid with coordinates X and Y running from 1 to 9.

**Usage**

`data(fires)`

**Format**

A data frame with 517 observations on the following 13 variables.

- `x` X coordinate for region, 0-10
- `y` Y coordinate for region, 0-10
- `month` an ordered factor with 12 levels
- `day` an ordered factor with 7 levels
- `ffmc` fine fuel moisture code
- `dmc` Duff moisture code
- `dc` drought code
- `isi` initial spread index
- `temp` average ambient temperature
- `RH` a numeric vector
- `wind` wind speed
- `rain` rainfall
- `burned` area burned in hectares

**Details**

This is the original data taken from the website below.

**Source**

References


Examples

```r
# Anova for month
summary(aov(burned~month, data=fires))
```

---

**gcv**

*Estimate EPE Using Delete-d Cross-Validation*

**Description**

This is a general purpose function to estimate the EPE of a specified cost function in regression and classification problems. For regression, the default cost function is for mean-square error and for classification it is the misclassification rate. Direct support for elastic penalty regression, LASSO, PCR, PLSR, nearest neighbour and Random Forest regression are included in the package. And for classification, built-in support functions are provided for LDA, QDA, Naive Bayes, kNN, CART, C5.0, Random Forest and SVM. Examples included in vignette section are provided for SCAD, MCP and best subset regression. Illustrative example datasets and data generation models are also provided.

**Usage**

```r
gcv(X, y, MaxIter = 1000, d = ceiling(length(y)/10), NCores = 1, 
cost = mse, yhat = yhat_lm, libs = character(0), seed = "default",
...)
```

**Arguments**

- `X`: inputs, matrix or dataframe
- `y`: output vector
- `MaxIter`: Number of iterations of the CV procedure
- `d`: Number of observations for the hold-out sample
- `NCores`: Default is 1 which does not use the parallel package. Otherwise, you can set to the number of cores available. If unsure, just experiment!
- `cost`: Average cost. See examples mse, mae, mape.
- `yhat`: In general it must be a function with arguments dfTrain and dfTest. See examples below.
- `libs`: Required libraries needed for the predictor.
- `seed`: Default is to use R’s default which is based on the current time. Otherwise set to an integer value. See Details.
- `...`: Additional arguments that are passed to yhat.
Details

If only serial evaluation was implemented then I would have used set.seed to control the random. But I have included it as an argument since it can be used to set the parallel random number generator seed. This is sometimes useful for replicating the simulations. If the argument seed is used, it will also set the seed when only serial computation is done.

Value

Matrix with one row and four columns: epe, sd_epe, snr, pcorr. These are respectively the estimated EPE, standard deviation of this estimate, an estimate of the snr (signal-to-noise ratio) out-of-sample and an out-of-sample estimate of the correlation between the prediction and the true value.

Note

The statistical distribution of the EPE’s when the argument outAllQ is set to TRUE is often positively skewed. This may be of interest in applications.

Author(s)

A. I. McLeod

References

ESL

See Also

mse, mae, mape, misclassificationrate, logloss, yhat_lm, yhat_nn, yhat_lars, yhat_plus, yhat_gel, yhat_step, yh_lda, yh_qda, yh_svm, yh_NB, yh_RF, yh_CART, yh_C50, yh_kNN, featureSelect, cv.glm

Examples

# Simple example but in general, MaxIter >= 1000 is recommended.
Xy <- Shaoreg()
gcv(Xy[,1:8], Xy[,9], MaxIter=25, d=5)

kNN_LOOCV

Select k with Leave-one-out CV

Description

Use leave-one-out CV to select k

Usage

kNN_LOOCV(X, y, kmax=ceiling(length(y)*0.5), plot=FALSE)
Arguments

- **x**: design matrix
- **y**: response vector
- **kmax**: maximum value of k to consider
- **plot**: show plot of mis-classification rate

Details

Leave one out CV is used for odd values of k from 1 to kmax.

Value

plot produced

Examples

```r
Xy <- rmix(300) # training data
kNN_LOOCV(Xy[,1:2], Xy[,3], plot=FALSE)
```

---

**kNN_MLE**

*MLE k in kNN*

Description

Uses the profile pseudolikelihood to obtain the estimate for k, the number of nearest neighbors parameter in kNN.

Usage

```r
kNN_MLE(X, Y, kmax = ceiling(length(Y) * 0.5), plot = TRUE)
```

Arguments

- **x**: An n-by-p matrix of covariates
- **y**: Outputs with Q classes
- **kmax**: The maximum size of k
- **plot**: if TRUE, plot the profile deviance otherwise no plot

Details

When Q=2, the glm algorithm is used to compute the profile pseudologlikelihood and for Q>2, the function multinom in nnet is used.
Value

The estimate of k obtained by maximizing the pseudolikelihood is returned. It can take any value from k=0 to k=kmax.

The result is returned invisibly if plot is TRUE.

Author(s)

A. I. McLeod Maintainer: <aimcleod@uwo.ca>

References


See Also

multinom

Examples

# Two classes example
X <- MASS::synth.tr[,1:2]
Y <- MASS::synth.tr[,3]
kNN_MLE(X=X, Y=Y, plot=FALSE)

## Not run:
# Three classes example
library("MASS") # need lda
Y<- iris[,5]
X<- iris[,1:4]
kopt <- kNN_MLE(X, Y)
kopt
# Mis-classification rates on training data.
# Of course FLDA does better in this case.
y <- factor(Y)
ans <- class::knn(train=X, test=X, k=kopt, cl=y)
etakNN <- sum(ans!=y)/length(y)
iris.ldf <- MASS::lda(X, y)
yfitFLDA <- MASS::predict.lda(iris.ldf, newdata=X, dimen=1)$class
eyFLDA <- sum(yfitFLDA!=y)/length(y)
etac <- etatFLDA, etakNN)
names(eta)<-c("FLDA", "kNN")
etal

## End(Not run)
**kyphosis**  
*Data on Children who have had Corrective Spinal Surgery*

**Description**

The kyphosis data frame has 81 rows and 4 columns, representing data on children who have had corrective spinal surgery.

**Usage**

`kyphosis`

**Format**

This data frame contains the following columns:

- **Kyphosis**  a factor with levels `absent` `present` indicating if a kyphosis (a type of deformation) was present after the operation.
- **Age**  in months
- **Number**  the number of vertebrae involved
- **Start**  the number of the first (topmost) vertebra operated on.

**Source**


**Examples**

```r
library("rpart")
fit <- rpart::rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
```

---

**logloss**  
*log-loss function for multiclass prediction*

**Description**

Cross entropy or logloss is computed.

**Usage**

`logloss(y, yp)`
**mae**

**Arguments**
- y: vector of test cases
- yp: corresponding vector of predictions

**Value**
- the log-loss

**Author(s)**
- A. I. McLeod

**References**
- Log loss is used in Kaggle competitions.

**See Also**
- misclassificationrate

**Examples**

```r
#logloss for perfect fit
  t <- ifelse(runif(50)<0.5, "a", "b")
  logloss(y=t, yp=t)
```

---

**mae**

*Mean Absolute Error*

**Description**

This is a widely used criterion since the time of Laplace. Just as least-squares is optimal for mean-square error loss functions, least absolute deviation is optimal for mean absolute error loss functions. See Wikipedia article [https://en.wikipedia.org/wiki/Mean_absolute_error](https://en.wikipedia.org/wiki/Mean_absolute_error).

**Usage**

```r
mae(yTest, yHat)
```

**Arguments**
- yTest: test data
- yHat: predictions of the test data

**Details**
- tba
Value

mean percentage absolute errors

Author(s)

A. I. McLeod

See Also

gcv, mse, mape, smape

Examples

mape(abs(rnorm(10)), rep(sqrt(2/pi),10))

description

This criterion is frequently used in business forecasting. See Wikipedia article https://en.wikipedia.org/wiki/Mean_absolute_percentage_error.

Usage

mape(yTest, yHat)

Arguments

yTest test data
yHat predictions of the test data

Details

tba

Value

mean percentage absolute errors

Author(s)

A. I. McLeod

See Also

gcv, mse, mae, smape
meatspec

Examples

\[ E(Z) = \sqrt{2/\pi}, \ \text{Z} - \text{abs}(N(0,1)) \]
\[ \text{mape(abs(rnorm(10)), rep(sqrt(2/\pi),10))} \]

---

**Meat Spectrometry to Determine Fat Content**

**Description**

A Tecator Infratec Food and Feed Analyzer working in the wavelength range 850 - 1050 nm by the Near Infrared Transmission (NIT) principle was used to collect data on samples of finely chopped pure meat. 215 samples were measured. For each sample, the fat content was measured along with a 100 channel spectrum of absorbances. Since determining the fat content via analytical chemistry is time consuming we would like to build a model to predict the fat content of new samples using the 100 absorbances which can be measured more easily.

**Usage**

data(meatspec)

**Format**

Dataset contains the following variables

- \( \text{V1-V100} \) absorbances across a range of 100 wavelengths
- \( \text{fat} \) fat content

**Source**

This data was used in Faraway’s book on Regression and his R package. He cites the following: H. H. Thodberg (1993) "Ace of Bayes: Application of Neural Networks With Pruning", report no. 1132E, Maglegaardvej 2, DK-4000 Roskilde, Danmark

---

**misclassificationrate**  *Misclassification Rate for Class Prediction*

**Description**

The misclassification rate is appropriate for 0-1 loss function for class prediction.

**Usage**

misclassificationrate(y, yp)
Arguments

y  vector of test cases
yp  corresponding vector of predictions

Value

the misclassification rate

Author(s)

A. I. McLeod

See Also

logloss

Examples

y <- c(3,1,1,3,3)
yh <- c(1,1,1,1,1)
misclassificationrate(y, yh)

mse

Mean Square Error Loss

Description

This is the default.

Usage

mse(yTest, yHat)

Arguments

yTest  test data
yHat  predictions of the test data

Details

 tba

Value

mean percentage absolute errors
pollution

Author(s)
A. I. McLeod

See Also
gcv, mae, mape, smape

Examples

def mean(abs(rnorm(10)), rep(sqrt(2/pi), 10))

<table>
<thead>
<tr>
<th>pollution</th>
<th>Pollution Data from McDonald and Schwing</th>
</tr>
</thead>
</table>

Description

The total age adjusted mortality rate, our response variable, for the years 1959-1961. The data from the U.S. covers 201 Standard Metropolitan Statistical Areas (SMSA).

Usage

data("pollution")

Format

A data frame with 60 observations on the following 16 variables.

- **prec**: Average annual precipitation in inches
- **jant**: Average January temperature in degrees F
- **jult**: Average July temperature in degrees F
- **ovrVU**: Percent of 1960 SMSA population aged 65 or older
- **popn**: Average household size
- **educ**: Median school years completed by those over 22
- **hous**: Percent of housing units which are sound and with all facilities
- **dens**: Population per sq. mile in urbanized areas, 1960
- **nonw**: Percent non-white population in urbanized areas, 1960
- **wwdrk**: Percent employed in white collar occupations
- **poor**: Percent of families with income less than 3000 USD
- **hc**: Relative hydrocarbon pollution potential
- **nox**: Relative nitric oxides pollution potential
- **sox**: Relative sulphur pollution potential
- **humid**: Annual average percent relative humidity at 1pm
- **mort**: Total age-adjusted mortality rate per 100,000
**References**

1973 Technometrics paper by McDonald and Schwing

**Examples**

```r
data(pollution)
str(pollution)
```

---

### prostate

**Prostate Cancer Data**

**Description**

Data to examine the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy.

**Usage**

```r
data(prostate)
```

**Format**

A data frame with 97 observations on the following 10 variables.

- `lcavol` log cancer volume
- `lweight` log prostate weight
- `age` in years
- `lbph` log of the amount of benign prostatic hyperplasia
- `svi` seminal vesicle invasion
- `lcp` log of capsular penetration
- `gleason` a numeric vector
- `pgg45` percent of Gleason score 4 or 5
- `lpsa` response

**Source**


**Examples**

```r
str(prostate)
```
Description

BFOS suggested this is data generation model for testing the performance of nonlinear classifiers such as CART. See details and vignette.

Usage

```r
digitsBFOS(n, eta = 0.25, alpha = NULL, silent = FALSE)
```

Arguments

- `n` Number of 10-tuples to generated.
- `eta` Bayes optimal missclassification rate.
- `alpha` Default is Null but if specified it is the probability line segment is flipped. When `alpha` is specified corresponding the Bayes rate is determined and shown.
- `silent` Default is `FALSE` and in this case the title is displayed otherwise no display.

Details

Breiman et al. (1984, Section 2.6.1, p.43) mentioned the case `alpha=0.1` and stated that the Bayes optimal rule has a 0.26 mis-classification rate. Derivation of this and more details are discussed in the vignette.

Value

A dataframe with 10*n rows and 8 columns is produced. Columns 1 to 7 are labeled `x1`, ..., `x7` and correspond to the inputs which are the line segments comprising each digit where 1 indicates on and 0 off. Column 8 is a factor with value the digit, 0, 1, ..., 9. Each successive block of ten rows corresponds to ten successive digits.

Note

An attribute "title" is created.

Author(s)

A. I. McLeod

References

BFOS (Breiman, Friedman, Olshen, and Stone), 1984 Classification and Regression Trees

See Also

`rxor`, `rmix`, `ShaoReg`
Examples

```r
#debug-rdigitsBFOS.R
#with alpha=0.1, not significantly different from 0.25
require("C50")
n <- 1000
Xy <- rdigitsBFOS(n, alpha=0.1)
attr(Xy, "title")
names(Xy)
ans <- C5.0(digit-., data=Xy)
XyTest <- rdigitsBFOS(n, alpha=0.1)
yHat <- predict(ans, newdata=XyTest[,1:7])
etta <- mean(yHat!=[YyTest$digit)
MOE95pc <- 1.96*sqrt(eta*(1-eta)/(10+n))
round(100*unlist(list(misclassificationRate=eta, "95pcMOE"=MOE95pc)),1)
```

regal  

Regression EPE for All Implemented Methods

Description

Determine EPE for many regression methods.

Usage

```r
regal(X, y, MaxIter = 1000, d = "default", Ncores = 1, plotBest = 6,
verboseQ = FALSE)
```

Arguments

- **X**  
  input matrix of dimension n-by-p with p<n
- **y**  
  output vector
- **MaxIter**  
  Number of CV iterations.
- **d**  
  Size of hold-out sample.
- **Ncores**  
  Number of cores to use for parallel processing.
- **plotBest**  
  Number of EPE’s to include on plot
- **verboseQ**  
  True, display progress, otherwise silent. When running R in Windows, the usual default is output buffering which means you will not see the extra output generated from verboseQ=TRUE until after regal() has finished. To see the output while this function is running you need to turn output buffering off. This can be done with the short-cut Ctrl-W. Another way to do this is to use the R Gui. Select Misc and the click on buffered output.

Value

A barplot is produced and matrix returned with rows corresponding to method and columns containing EPE, sd(EPE), snr and two correlation estimates between forecast and true value.
**rmix**

*Random Mixture Classification Example*

**Description**

Generates a random mixture for binary class prediction with output variable green and red factors and with two inputs x1 and x2. Similar to the mixture dataset in ESL.

**Usage**

```r
rmix(n = 100)
```

**Arguments**

- `n` Sample size, should be even number, n/2 green and n/2 red.

**Details**

The optimal Bayes error rate is 20.76

**Value**

dataframe with columns x1, x2, y

**Author(s)**

A. I. McLeod
**References**


**See Also**

rxor, rdigitsBFOS, ShaoReg

**Examples**

```r
mdf <- rmix(200)
gr <- mdf[mdf$y=="green",]
rd <- mdf[mdf$y=="red",]
with(mdf, {plot(x1, x2, type="n")
  points(gr, col="green")
  points(rd, col="red")
})
```

**Description**

Data is generated for the XOR problem. The default settings produce a data.frame with columns x1, x2 and y and with 4 rows and this table defines the XOR problem. The output y is defined by the XOR operation applied to the Boolean x1 and x2.

**Usage**

```r
rxor(n = 1, p = 0)
```

**Arguments**

- `n` sample size is 4*n
- `p` extra random inputs, x3, x4, etc. So the output data frame has dimensions 4*n by 2+p+1 columns. The extra p columns are random Bernouilli random variables with equi-probable outcomes.

**Details**

This was a famous problem in online learning.

**Value**

data.frame with 4*n rows and 2+p+1 columns. The last column corresponds to the output.

**Author(s)**

A. I. McLeod
Examples

```r
library("C50")
Xy <- rxor(n=5, p=2)
C5.0(y ~ ., data=Xy)
```

---

### Description

Simulated multiple linear regression data from a model used in simulation experiments reported in Shao’s famous paper on cross-validation for model selection.

### Usage

```r
ShaoReg(n = 20, beta = c(3, 1.5, 0, 0, 2, 0, 0, 0), rho = 0.5, sig = 1)
```

### Arguments

- `n`: sample size, length of output
- `beta`: regression coefficients
- `rho`: cross-covariance, must be less than in magnitude 1
- `sig`: residual standard deviation

### Details

In general the regression equation used for simulation is:

$$ y = X\beta + \epsilon $$

where $\beta$ is a vector of the regression coefficients of length $p$, $X$ is the design matrix with $n$ rows and $p$ columns and $\epsilon$ is a vector of $n$ independent normal random variables with mean zero and standard deviation $\sigma$. The rows of $X$ are $p$-variate normal with mean vector zero and $p$-by-$p$ covariance matrix $(i,j)$-entry $\rho|i-j|$.

Shao (1993) used the default settings in the arguments and $n = 20, 60, 100$ in simulation experiments with delete-d cross-validation.

### Value

Data frame with $n$ rows and $p+1$ columns. The first $p$ columns are labelled `x1`, ..., `xp` and the last column is `y`.

### Author(s)

A. I. McLeod
References


Examples

ShaoReg()

---

SinghTest

Singh Prostate Microarray Test Data

Description

Microarray data for 25 prostate tumors and 9 nontumors from patients undergoing surgery.

Usage

data("SinghTest")

Format

A data frame with 102 observations on the following 101 variables.

gene1  a numeric vector
gene2  a numeric vector
gene3  a numeric vector
gene4  a numeric vector
gene5  a numeric vector
gene6  a numeric vector
gene7  a numeric vector
gene8  a numeric vector
gene9  a numeric vector
gene10 a numeric vector
gene11 a numeric vector
gene12 a numeric vector
gene13 a numeric vector
gene14 a numeric vector
gene15 a numeric vector
gene16 a numeric vector
gene17 a numeric vector
gene18 a numeric vector
gene19 a numeric vector
| gene20   | a numeric vector |
| gene21   | a numeric vector |
| gene22   | a numeric vector |
| gene23   | a numeric vector |
| gene24   | a numeric vector |
| gene25   | a numeric vector |
| gene26   | a numeric vector |
| gene27   | a numeric vector |
| gene28   | a numeric vector |
| gene29   | a numeric vector |
| gene30   | a numeric vector |
| gene31   | a numeric vector |
| gene32   | a numeric vector |
| gene33   | a numeric vector |
| gene34   | a numeric vector |
| gene35   | a numeric vector |
| gene36   | a numeric vector |
| gene37   | a numeric vector |
| gene38   | a numeric vector |
| gene39   | a numeric vector |
| gene40   | a numeric vector |
| gene41   | a numeric vector |
| gene42   | a numeric vector |
| gene43   | a numeric vector |
| gene44   | a numeric vector |
| gene45   | a numeric vector |
| gene46   | a numeric vector |
| gene47   | a numeric vector |
| gene48   | a numeric vector |
| gene49   | a numeric vector |
| gene50   | a numeric vector |
| gene51   | a numeric vector |
| gene52   | a numeric vector |
| gene53   | a numeric vector |
| gene54   | a numeric vector |
| gene55   | a numeric vector |
| gene56   | a numeric vector |
gene57  a numeric vector
gene58  a numeric vector
gene59  a numeric vector
gene60  a numeric vector
gene61  a numeric vector
gene62  a numeric vector
gene63  a numeric vector
gene64  a numeric vector
gene65  a numeric vector
gene66  a numeric vector
gene67  a numeric vector
gene68  a numeric vector
gene69  a numeric vector
gene70  a numeric vector
gene71  a numeric vector
gene72  a numeric vector
gene73  a numeric vector
gene74  a numeric vector
gene75  a numeric vector
gene76  a numeric vector
gene77  a numeric vector
gene78  a numeric vector
gene79  a numeric vector
gene80  a numeric vector
gene81  a numeric vector
gene82  a numeric vector
gene83  a numeric vector
gene84  a numeric vector
gene85  a numeric vector
gene86  a numeric vector
gene87  a numeric vector
gene88  a numeric vector
gene89  a numeric vector
gene90  a numeric vector
gene91  a numeric vector
gene92  a numeric vector
gene93  a numeric vector
gene94  a numeric vector
gene95  a numeric vector
gene96  a numeric vector
gene97  a numeric vector
gene98  a numeric vector
gene99  a numeric vector
gene100 a numeric vector
health  a factor with levels normal tumor

Details
The data have been standardized by patient. The best 100 genes out of 12600 genes in the original
have been selected. Pochet et al. (2004) suggested this test dataset. It was also mentioned in Speed’s
book.

Source
benchmarking of microarray data classification: assessing the role of nonlinearity and dimension-

References
Terry Speed

See Also
featureSelect, churnTrain

Examples
require("MASS")
data(SinghTest)
BestGenes <- 10
XTr <- SinghTrain[,1:BestGenes]
yTr <- SinghTrain$health
ans <- lda(x=XTr, grouping=yTr)
XTe <- SinghTest[,1:BestGenes]
yH <- predict(ans, newdata=XTe)$class
yTe <- SinghTest$health
table(yTe, yH)
SinghTrain

Singh Prostate Microarray Training Data

Description
Microarray data for 52 prostate tumors and 50 nontumors from patients undergoing surgery.

Usage
data("SinghTrain")

Format
A data frame with 102 observations on the following 101 variables.
gene1 a numeric vector
gene2 a numeric vector
gene3 a numeric vector
gene4 a numeric vector
gene5 a numeric vector
gene6 a numeric vector
gene7 a numeric vector
gene8 a numeric vector
gene9 a numeric vector
gene10 a numeric vector
gene11 a numeric vector
gene12 a numeric vector
gene13 a numeric vector
gene14 a numeric vector
gene15 a numeric vector
gene16 a numeric vector
gene17 a numeric vector
gene18 a numeric vector
gene19 a numeric vector
gene20 a numeric vector
gene21 a numeric vector
gene22 a numeric vector
gene23 a numeric vector
gene24 a numeric vector
gene25 a numeric vector
gene26  a numeric vector
gene27  a numeric vector
gene28  a numeric vector
gene29  a numeric vector
gene30  a numeric vector
gene31  a numeric vector
gene32  a numeric vector
gene33  a numeric vector
gene34  a numeric vector
gene35  a numeric vector
gene36  a numeric vector
gene37  a numeric vector
gene38  a numeric vector
gene39  a numeric vector
gene40  a numeric vector
gene41  a numeric vector
gene42  a numeric vector
gene43  a numeric vector
gene44  a numeric vector
gene45  a numeric vector
gene46  a numeric vector
gene47  a numeric vector
gene48  a numeric vector
gene49  a numeric vector
gene50  a numeric vector
gene51  a numeric vector
gene52  a numeric vector
gene53  a numeric vector
gene54  a numeric vector
gene55  a numeric vector
gene56  a numeric vector
gene57  a numeric vector
gene58  a numeric vector
gene59  a numeric vector
gene60  a numeric vector
gene61  a numeric vector
gene62  a numeric vector
gene63  a numeric vector
gene64  a numeric vector
gene65  a numeric vector
gene66  a numeric vector
gene67  a numeric vector
gene68  a numeric vector
gene69  a numeric vector
gene70  a numeric vector
gene71  a numeric vector
gene72  a numeric vector
gene73  a numeric vector
gene74  a numeric vector
gene75  a numeric vector
gene76  a numeric vector
gene77  a numeric vector
gene78  a numeric vector
gene79  a numeric vector
gene80  a numeric vector
gene81  a numeric vector
gene82  a numeric vector
gene83  a numeric vector
gene84  a numeric vector
gene85  a numeric vector
gene86  a numeric vector
gene87  a numeric vector
gene88  a numeric vector
gene89  a numeric vector
gene90  a numeric vector
gene91  a numeric vector
gene92  a numeric vector
gene93  a numeric vector
gene94  a numeric vector
gene95  a numeric vector
gene96  a numeric vector
gene97  a numeric vector
gene98  a numeric vector
gene99  a numeric vector
gene100 a numeric vector

health  a factor with levels normal tumor
Details
The data have been standardized by patient. The best 100 genes out of 12600 genes in the original have been selected.

Source

References
Terry Speed

See Also
featureSelect, churnTrain

Examples

```r
yh.C50(SinghTrain, SinghTest)
```

```r
dim(SinghTrain)
dim(SinghTest)
```

<table>
<thead>
<tr>
<th>smape</th>
<th>Mean Absolute Percentage Error</th>
</tr>
</thead>
</table>

Description
This criterion is frequently used in business forecasting. See Wikipedia article [https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error](https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error).

Usage

```r
smape(yTest, yHat)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yTest</td>
<td>test data</td>
</tr>
<tr>
<td>yHat</td>
<td>predictions of the test data</td>
</tr>
</tbody>
</table>

Details

<table>
<thead>
<tr>
<th>Details</th>
<th>tba</th>
</tr>
</thead>
</table>

Value
mean percentage absolute errors
Author(s)

A. I. McLeod

See Also
gcv, mse, mape

Examples

smape(abs(rnorm(10)), rep(sqrt(2/pi),10))

data(longley)
vifx(longley[,1:6])

Description

Variance inflation factor is computed for a given regression design matrix.

Usage

vifx(X)

Arguments

X Design matrix, should include column of 1’s if there is an intercept term.

Details

The design matrix is assumed to be of full rank.

Value

the variance inflation factors for each column

Author(s)

A. I. McLeod

Examples

data(longley)
vifx(longley[,1:6])
Description

Regression prediction using tree method.

Usage

\texttt{yhat\_CART(dfTrain, dfTest)}

Arguments

- \texttt{dfTrain} : Data frame for training data. Last column must be the output variable.
- \texttt{dfTest} : Data frame for test data. Last column must be the output variable.

Value

The predictions for the test sample

Author(s)

A. I. McLeod

Examples

\begin{verbatim}
x <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
ite <- sample(n, size=d)
itr <- (1:n)[!(match(1:n, ite, nomatch = 0) > 0)]
trdf <- data.frame(X[itr,], y=y[itr]) # X, y already defined
tedf <- data.frame(X[ite,], y=y[ite])
yhat\_CART(trdf, tedf)
\end{verbatim}
yhat_gel  

**Elastic Net Regression Prediction**

**Description**

Fit regression using 10-fold CV with the 1 standard deviation rule and compute predictions.

**Usage**

```r
yhat_gel(dfTrain, dfTest, alpha = 1)
```

**Arguments**

- `dfTrain`: Data frame for training data. Last column must be the output variable.
- `dfTest`: Data frame for test data. Last column must be the output variable.
- `alpha`: Must be in [0,1], alpha=1 for LASSO (default), alpha=0 for ridge regression. Another recommended choice is alpha=0.5.

**Value**

The predictions for the test sample

**Author(s)**

A. I. McLeod

**Examples**

```r
Xy <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size=d)
iTr <- (1:n)[!match(1:n, iTe, nomatch = 0) > 0]
trdf <- data.frame(X[iTr, ], y=y[iTr]) # X, y already defined
tedf <- data.frame(X[iTe, ], y=y[iTe])
yhat_gel(trdf, tedf)
```
**yhat_lars**  

**Fit LASSO Regression using Mallows Cp and Predict**

**Description**

LASSO regression is fit using the lars algorithm.

**Usage**

```r
yhat_lars(dfTrain, dfTest, normalize = TRUE)
```

**Arguments**

- `dfTrain`: Data frame for training data. Last column must be the output variable.
- `dfTest`: Data frame for test data. Last column must be the output variable.
- `normalize`: Default TRUE means the predictors are centered and scaled. Otherwise no transformation.

**Value**

The predictions for the test sample.

**Author(s)**

A. I. McLeod

**Examples**

```r
Xy <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size=d)
iTr <- (1:n)[!match(1:n, iTe, nomatch = 0) > 0]
trdf <- data.frame(X[iTr, ], y=y[iTr]) #X, y already defined
tedf <- data.frame(X[iTe, ], y=y[iTe])
yhat_lars(trdf, tedf)
```
**yhat_lm**

*Linear Predictor using Least-Squares Regression*

**Description**

This is the default predictor used by getEPE and is provided as an example.

**Usage**

```r
yhat_lm(dfTrain, dfTest)
```

**Arguments**

- `dfTrain` Data frame for training data. Last column must be the output variable.
- `dfTest` Data frame for test data. Last column must be the output variable.

**Value**

The predictions for the test sample

**Author(s)**

A. I. McLeod

**Examples**

```r
Xy <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size=d)
iTr <- (1:n)[!match(1:n, iTe, nomatch = 0) > 0]
trdf <- data.frame(X[iTr,], y=y[iTr]) #X, y already defined
tedf <- data.frame(X[iTe,], y=y[iTe])
yhat_lm(trdf, tedf)
```
Description

Nearest neighbour prediction

Usage

\[ \text{yhat.nn}(\text{dfTrain}, \text{dfTest}, \text{normalize} = \text{TRUE}) \]

Arguments

- \text{dfTrain}: Data frame for training data. Last column must be the output variable.
- \text{dfTest}: Data frame for test data. Last column must be the output variable.
- \text{normalize}: Default TRUE means the predictors are centered and scaled. Otherwise no transformation.

Value

The predictions for the test sample

Author(s)

A. I. McLeod

Examples

\[
\begin{align*}
X & \leftarrow \text{prostate} \\
X & \leftarrow \text{prostate[, -9]} \\
y & \leftarrow \text{prostate[, 9]} \\
n & \leftarrow \text{length}(y) \\
d & \leftarrow 10 \\
\text{set.seed}(777513) \\
\text{iTe} & \leftarrow \text{sample}(n, \text{size}=d) \\
\text{iTr} & \leftarrow \text{!(iTe)} \leftarrow n \leftarrow \text{match}(1:n, \text{iTe, nomatch} = \text{0}) > 0) > 0 \\
\text{trdf} & \leftarrow \text{data.frame}(X[iTr,], y=y[iTr]) \ # X, y already defined \\
\text{tedf} & \leftarrow \text{data.frame}(X[iTe,], y=y[iTe]) \\
\text{yhat.nn}(\text{trdf}, \text{tedf})
\end{align*}
\]
SCAD or MCP Regression Prediction

Description

Fits penalized regression with SCAD or MCP penalty and computes the predictions for the test data.

Usage

```r
yhat_plus(dfTrain, dfTest, normalize = TRUE, ic = c("BIC", "AIC"),
method = c("scad", "mc+", "lasso"))
```

Arguments

- `dfTrain`: Data frame for training data. Last column must be the output variable.
- `dfTest`: Data frame for test data. Last column must be the output variable.
- `normalize`: Default TRUE means the predictors are centered and scaled. Otherwise no transformation.
- `ic`: "AIC" or "BIC"
- `method`: "scad", "mc+" or "lasso"

Value

The predictions for the test sample

Author(s)

A. I. McLeod

Examples

```r
Xy <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size=d)
iTr <- (1:n)[!match(1:n, iTe, nomatch = 0) > 0]
trdf <- data.frame(X[iTr,], y=y[iTr]) #X, y already defined
tedf <- data.frame(X[iTe,], y=y[iTe])
yhat_plus(trdf, tedf)
```
yhat_RF

Fit Random Forest Regression Predictor

Description

Random Forest prediction on test data

Usage

yhat_RF(dfTrain, dfTest)

Arguments

dfTrain Data frame for training data. Last column must be the output variable.
dfTest Data frame for test data. Last column must be the output variable.

Value

The predictions for the test sample

Author(s)

A. I. McLeod

Examples

```r
Xy <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size=d)
iTr <- 1:1:length(iTe)[match(1:1:length(iTe), y)] # y already defined
trdf <- data.frame(X[iTr,], y=y[iTr])
tedf <- data.frame(X[iTe,], y=y[iTe])
yhat_plus(trdf, tedf)
```
**Description**

Fits a subset regression model using backward stagewise regression to training data and computes the predictions for the test data.

**Usage**

`yhat_step(dfTrain, dfTest, ic = c("BIC", "AIC"))`

**Arguments**

- `dfTrain` Data frame for training data. Last column must be the output variable.
- `dfTest` Data frame for test data. Last column must be the output variable.
- `ic` Information criterion to use to select the number of components. Default is BIC.

**Value**

The predictions for the test sample

**Author(s)**

A. I. McLeod

**Examples**

```r
xy <- prostate
X <- prostate[, -9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size=d)
iTr <- (1:n)[!match(1:n, iTe, nomatch = 0) > 0]
trdf <- data.frame(X[iTr, ], y=y[iTr]) #X, y already defined
tedf <- data.frame(X[iTe, ], y=y[iTe])
yhat_step(trdf, tedf)
```
**yhat_SVM**

_Support Vector Machine Regression Prediction_

### Description

SVM prediction on test data

### Usage

```r
yhat_SVM(dfTrain, dfTest)
```

### Arguments

- `dfTrain`  
  Data frame for training data. Last column must be the output variable.

- `dfTest`  
  Data frame for test data. Last column must be the output variable.

### Value

The predictions for the test sample

### Author(s)

A. I. McLeod

### Examples

```r
Xy <- prostate
X <- prostate[, 9]
y <- prostate[, 9]
n <- length(y)
d <- 10
set.seed(777513)
iTe <- sample(n, size = d)
iTr <- (1:n)[!match(1:n, iTe, nomatch = FALSE) > 0]
trdf <- data.frame(X[iTr, ], y = y[iTr])  # X, y already defined
tedf <- data.frame(X[iTe, ], y = y[iTe])
yhat_SVM(trdf, tedf)
```
Description
Given training data and test examples, the C50 predictions for the test data are produced and the
misclassification rate is returned.

Usage
yh_C50(dfTr, dfTe)

Arguments
dfTr dataframe with last column for the output. The output must be a factor.
dfTe dataframe for test data. Must have columns corresponding to the training columns
except the test output is not needed.

Value
tba. Not fully implemented yet.

Author(s)
A. I. McLeod

See Also
yh_CART, yh_RF, yh_svm, yh_NB, yh_kNN, yh_lda, yh_logistic, yh_qda

Examples
yh_C50(SinghTrain, SinghTest)#0.235

Description
Given training data and test examples, the CART predictions for the test data are produced and the
misclassification rate is returned.

Usage
yh_CART(dfTr, dfTe)
Arguments

dfTr    dataframe with last column for the output. The output must be a factor.
dfTe    dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.

Value

tba. Not fully implemented yet.

Author(s)

A. I. McLeod

See Also

yh_C50, yh_RF, yh_svm, yh_NB, yh_kNN, yh_ldap, yh_logistic, yh_qda

Examples

```
yh_CART(SinghTrain, SinghTest) #0.32
```

Description

Given training data and test examples, the kNN predictions for the test data are produced. The tuning parameter k is automatically selected by specifying one of the methods: LOOCV, MLE or NN.

Usage

```
yh_knn(dfTr, dfTe, method = c("LOOCV", "MLE", "NN"), k=1)
```

Arguments

- dfTr: dataframe with last column for the output. The output must be a factor.
- dfTe: dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.
- method: One of the automatic methods for selecting k, the number of nearest neighbours. The default is LOOCV.
- k: Pre-specified k but this value of k is only used when method="NN" otherwise when method="LOOCV" or method="MLE", k is estimated.

Value

The mis-classification rate (cost) and correlation of prediction and test.
Author(s)
A. I. McLeod

See Also
yh_C50, yh_CART, yh_RF, yh_svm, yh_NB, yh_lda, yh_logistic, yh_qda

Examples

```r
yh_kNN(SinghTrain[,c(1:10, 101)], SinghTest[,c(1:10, 101)])#0.088
yh_kNN(SinghTrain[,c(1:10, 101)], SinghTest[,c(1:10, 101)], method="NN")#0.088
```

---

**yh_lda**  
*LDA predictions*

Description
Given training data and test examples, the LDA predictions for the test data are produced.

Usage

```
yh_lda(dfTr, dfTe)
```

Arguments

- `dfTr`: dataframe with last column for the output. The output must be a factor.
- `dfTe`: dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.

Value

tba. Not fully implemented yet.

Author(s)
A. I. McLeod

Examples

```r
library("MASS")
data(SinghTest) # is 0
yh_lda(SinghTrain[,c(1:10, 101)], SinghTest[,c(1:10, 101)])
```
Description

The training data is fit and then the mis-classification rate for the test data is computed.

Usage

\[
yh\_logistic(df\_Tr, df\_Te, alpha = NULL)
\]

Arguments

- \( df\_Tr \): Training data frame, last column factor response and other columns are numeric inputs.
- \( df\_Te \): Test data frame, columns same variables as in training data frame
- \( alpha \): alpha=1 for LASSO, alpha=0.5 for half-mixture, alpha=0 for ridge regression

Details

alpha=0.02 often is numerically better behaved than alpha=0

Value

vector with named values misclassificationRate, logloss, pcorr

Author(s)

A. I. McLeod

Examples

\[
z \leftarrow \text{kyphosis[,c(2:4,1)]}
set.seed(37771)
i \leftarrow \text{sample(1:81, size=7, replace=TRUE)}
df\_Te \leftarrow z[i,]
i \leftarrow \text{setdiff(1:81, i)}
df\_Tr \leftarrow z[i,]
yh\_logistic(df\_Tr, df\_Te)
yh\_logistic(df\_Tr, df\_Te, alpha=1)
## Not run: #cross-validation, takes a few minutes
X \leftarrow \text{kyphosis[,3:4]}
y \leftarrow \text{kyphosis[,4]}
cgcv(X, y, yh=\text{yh\_logistic}, NCores=8)
cgcv(X, y, yh=\text{yh\_logistic}, NCores=8, alpha=1)
cgcv(X, y, yh=\text{yh\_logistic}, NCores=8, alpha=0.5)
cgcv(X, y, yh=\text{yh\_logistic}, NCores=8, alpha=0.02)
#
## End(Not run)
### yh_NB

**Naive Bayes Prediction**

**Description**

Given training data and test examples, the NB predictions for the test data are produced and the misclassification rate is returned.

**Usage**

```r
yh_NB(dfTr, dfTe)
```

**Arguments**

- `dfTr`: dataframe with last column for the output. The output must be a factor.
- `dfTe`: dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.

**Value**

tba. Not fully implemented yet.

**Author(s)**

A. I. McLeod

**Examples**

```r
yh_NB(SinghTrain, SinghTest)#0
```

### yh_qda

**QDA Prediction**

**Description**

Given training data and test examples, the QDA predictions for the test data are produced.

**Usage**

```r
yh_qda(dfTr, dfTe)
```

**Arguments**

- `dfTr`: dataframe with last column for the output. The output must be a factor.
- `dfTe`: dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.
**Description**

Given training data and test examples, the RF predictions for the test data are produced and the misclassification rate is returned.

**Usage**

```r
yh_RF(dfTr, dfTe)
```

**Arguments**

- **dfTr**: dataframe with last column for the output. The output must be a factor.
- **dfTe**: dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.

**Value**

tba. Not fully implemented yet.

**Author(s)**

A. I. McLeod

**Examples**

```r
yh_RF(SinghTrain, SinghTest)#0
```
**Description**

Given training and test examples, the SVM predictions for the test data are produced.

**Usage**

```r
yh_svm(dfTr, dfTe)
```

**Arguments**

- `dfTr`: dataframe with last column for the output. The output must be a factor.
- `dfTe`: dataframe for test data. Must have columns corresponding to the training columns except the test output is not needed.

**Value**

`tba. Not fully implemented yet.`

**Author(s)**

A. I. McLeod

**Examples**

```r
yh_svm(SinghTrain, SinghTest)#0.294
```
Index

*Topic **classif**
cgcv, 5
defeatureSelect, 10
gcv, 12
gencve-package, 2
kNN_LOOCV, 13
kNN_MLE, 14
miscalssificationrate, 19
rdigitsBFOS, 23
rmix, 25
rxor, 26
yh_C50, 46
yh_CART, 46
yh_kNN, 47
yh_1da, 48
yh_logistic, 49
yh_NB, 50
yh_qda, 50
yh_RF, 51
yh_svm, 52

*Topic **datagen**
rddigitsBFOS, 23
rmix, 25
rxor, 26
ShaoReg, 27

*Topic **datasets**
churn, 6
Detroit, 7
fires, 11
kyphosis, 16
meatspec, 19
pollution, 21
prostate, 22
SinghTest, 28
SinghTrain, 32

*Topic **models**
cgcv, 5
dShao, 9
defeatureSelect, 10
gcv, 12
gencve-package, 2
mae, 17
mape, 18
mse, 20
rdigitsBFOS, 23
regal, 24
rmix, 25
rxor, 26
ShaoReg, 27
smape, 35
vifx, 36
yh_C50, 46
yh_CART, 46
yh_kNN, 47
yh_1da, 48
yh_logistic, 49
yh_NB, 50
yh_qda, 50
yh_RF, 51
yh_svm, 52
yhat_CART, 37
yhat_gel, 38
yhat_lars, 39
yhat_lm, 40
yhat_nn, 41
yhat_plus, 42
yhat_step, 44

*Topic **package**
gencve-package, 2

*Topic **regression**
dShao, 9
defeatureSelect, 10
gcv, 12
gencve-package, 2
mae, 17
mape, 18
mse, 20
regal, 24
INDEX

ShaoReg, 27
smape, 35
vifx, 36
yhat_CART, 37
yhat_gel, 38
yhat_lars, 39
yhat_lm, 40
yhat_nn, 41
yhat_plus, 42
yhat_step, 44

SinghTest, 28
SinghTrain, 7, 32
smape, 18, 21, 35

vifx, 36

yh_C50, 13, 46, 47, 48
yh_CART, 13, 46, 46, 48
yh_kNN, 13, 46, 47, 47
yh_lda, 13, 46–48, 48
yh_logistic, 46–48, 49
yh_NB, 13, 46–48, 50
yh_NN (yh_kNN), 47
yh_qda, 13, 46–48, 50
yh_RF, 13, 46–48, 51
yh_svm, 13, 46–48, 52
yhat_CART, 37
yhat_gel, 13, 38
yhat_lars, 13, 39
yhat_lm, 13, 40
yhat_nn, 13, 41
yhat_plus, 13, 42
yhat_RF, 43
yhat_step, 13, 44
yhat_SVM, 45

cgcv, 5
churn, 6
churnTest (churn), 6
churnTrain, 31, 35
churnTrain(churn), 6
cv.glm, 5, 13

detroit, 7
dShao, 9

featureSelect, 7, 10, 13, 31, 35
fires, 11

gcv, 9, 12, 18, 21, 25, 36
genCve (genCve-package), 2
genCve-package, 2

kNN_LOOCV, 13
kNN_MLE, 14
kyphosis, 16

logloss, 13, 16, 20

mae, 13, 17, 18, 21
mape, 13, 18, 18, 21, 36
meatspec, 19
misclassificationrate, 13, 17, 19
mse, 13, 18, 20, 36
multinom, 15

pollution, 21
prostate, 22

rdigitsBFOS, 23, 26
regal, 24
rmix, 23, 25
rxor, 23, 26, 26

ShaoReg, 23, 26, 27