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:

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	Detroit Homicide Data for 1961-73
ShaoReg	Synthetic Regression Data
SinghTest	Singh Prostate Microarray Test Data
SinghTrain	Singh Prostate Microarray Training Data
cgcv	Estimate Misclassification Rate Using d-fold Cross-Validation for Class Prediction
churnTrain	Customer Churn Data
dShao	Shao Holdout Sample Size for Linear Regression Variable Selection
featureSelect	Feature Select For Wide Data
fires	Forest Fires in Montesinho Natural Park
gcv	Estimate EPE Using Delete-d Cross-Validation
gencve-package	General Cross Validation Engine General Cross Validation Engine
kNN_LOOCV	Select k with Leave-one-out CV
kNN_MLE	MLE k in kNN
kyphosis	Data on Children who have had Corrective Spinal Surgery
logloss	log-loss function for multiclass prediction
mae	Mean Absolute Error
mape	Mean Absolute Percentage Error
meatspec	Meat Spectrometry to Determine Fat Content
misclassificationrate	Misclassification Rate for Class Prediction
mse	Mean Square Error Loss
pollution	Pollution Data from McDonald and Schwing
prostate	Prostate Cancer Data
rdigitsBFOS	BFOS Digit Recognition Problem
regal	Regression EPE for All Implemented Methods
rmix	Random Mixture Classification Example
rxor	Random XOR Samples
smape	Mean Absolute Percentage Error
vifx	Variance Inflation Factor
yh_C50	C50 Prediction
yh_CART	CART Prediction
yh_NB	Naive Bayes Prediction
yh_RF	Random Forest Prediction

yh_kNN	kNN or NN prediction
yh_lda	LDA predictions
yh_logistic	Logistic Regression and Regularized Logistic Regression Prediction
yh_qda	QDA Prediction
yh_svm	Support Vector Machine Prediction
yhat_CART	CART regression prediction
yhat_RF	Fit Random Forest Regression Predictor
yhat_SVM	Support Vector Machine Regression Prediction
yhat_gel	Elastic Net Regression Prediction
yhat_lars	Fit LASSO Regression using Mallows Cp and Predict
yhat_lm	Linear Predictor using Least-Squares Regression
yhat_nn	Nearest Neighbour Prediction
yhat_plus	SCAD or MCP Regression Prediction
yhat_step	Backward Stagewise Regression with AIC or BIC

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 classifcation 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 regularizatin 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.

Jun Shao (1993), Linear Model Selection by Cross-validation Journal of the American Statistical Association,Vol. 88, Iss. 422, 1993.

J. H. Kim, (2009), Estimating Classification Error Rate: Repeated Cross-validation, Repeated Hold-out and Bootstrap. Computational Statistics and Data Analysis, 53, 3735-3745.

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 missclassification 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-classifcation rate

Author(s)

A. I. McLeod

Examples

```
Xy <- rmix(200) #training data
X <- as.matrix.data.frame(Xy[,1:2])
y <- Xy[,3]
cgcv(X, y, 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)
    international plan = no
    total day minutes <= 223.2
    number customer service calls <= 3
    ->  class 0  [0.973]

Rule 5: (1972/87, lift 1.1)
    total day minutes <= 264.4
    total intl minutes <= 13.1
    total intl calls > 2
    number customer service calls <= 3
    ->  class 0  [0.955]

Rule 10: (60, lift 6.8)
    international plan = yes
```

```
    total intl calls <= 2
    ->  class 1  [0.984]

Rule 12: (32, lift 6.7)
    total day minutes <= 120.5
    number customer service calls > 3
    ->  class 1  [0.971]
```

This implementation of C5.0 contains the same rules, but the rule numbers are different than above.

Usage

```
data(churn)
```

Value

churnTrain	The training set
churnTest	The test set.

Source

From CRAN package C50, copy is included for convenience.

See Also

featureSelect, SinghTrain

Detroit *Detroit Homicide Data for 1961-73*

Description

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.

Usage

```
data(Detroit)
```

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 orginally 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

Fisher, J.C. (1976). Homicide in Detroit: The Role of Firearms. Criminology, vol.14, 387-400.

Gunst, R.F. and Mason, R.L. (1980). Regression analysis and its application: A data-oriented approach. Marcel Dekker.

Miller, A. J. (2002). Subset Selection in Regression. 2nd Ed. Chapman & Hall/CRC. Boca Raton.

Examples

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

dShao *Shao Holdout Sample Size for Linear Regression Variable Selection*

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

Jun Shao (1993). Model Selection by Cross-Validation. Journal of the American Statistical Association, Vol. 88, No. 422, pp. 486-494.

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 indicies 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]
```

`fires` *Forest Fires in Montesinho Natural Park*

Description

The forest fire data were collected during January 2000 to December 2003 for fires in theMontesinho 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 as 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` X 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

http://archive.ics.uci.edu/ml/datasets/Forest+Fires

References

P. Cortez and A. Morais, 2007. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007.

Examples

```
#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

```
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

```
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

```
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

Holmes, C. C. and Adams, N. M. (2003). Likelihood inference in nearest-neighbour classification models, Biometrika, 90(1), 99-112. http://biomet.oxfordjournals.org/cgi/content/abstract/90/1/99

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
etaFLDA <- sum(yfitFLDA!=y)/length(y)
eta<-c(etaFLDA, etaKNN)
names(eta)<-c("FLDA", "kNN")
eta

## 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

John M. Chambers and Trevor J. Hastie eds. (1992) *Statistical Models in S*, Wadsworth and Brooks/Cole, Pacific Grove, CA.

Examples

```
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)
```

Arguments

y	vector of test cases
yp	corresonding vector of predictions

Value

the log-loss

Author(s)

A. I. McLeod

References

Log loss is used in Kaggle competitions.

See Also

misclassificationrate

Examples

```
#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.

Usage

```
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))
```

---

mape *Mean Absolute Percentage Error*

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

Examples

```
#E(Z)=sqrt(2/pi), Z~abs(N(0,1))
mape(abs(rnorm(10)), rep(sqrt(2/pi),10))
```

---

meatspec *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

`V1-V100` absorbances across a range of 100 wavelengths

`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	corresonding 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

Author(s)

A. I. McLeod

See Also

gcv, mae, mape, smape

Examples

```
mse(abs(rnorm(10)), rep(sqrt(2/pi),10))
```

---

pollution *Pollution Data from McDonald and Schwing*

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 January temperature in degrees F

OVR65 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

```
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

```
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

Stamey, T., Kabalin, J., McNeal, J., Johnstone, I., Freiha, F., Redwine, E. and Yang, N (1989) Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate II. Radical prostatectomy treted patients, Journall of Urology 16: 1076–1083.

Examples

```
str(prostate)
```

rdigitsBFOS *BFOS Digit Recognition Problem*

Description

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

Usage

```
rdigitsBFOS(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 theprobability 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

```
#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])
eta <- mean(yHat!=XyTest$digit)
MOE95pc <- 1.96*sqrt(eta*(1-eta)/(10*n))
round(100*unlist(list(misclassifcationRate=eta, "95pcMOE"=MOE95pc)),1)
```

`regal` *Regression EPE for All Implemented Methods*

Description

Determine EPE for many regression methods.

Usage

```
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.

Author(s)

A. I. McLeod

See Also

gcv

Examples

```
#about 200 seconds
## Not run:
  data(prostate)
  X <- as.matrix.data.frame(prostate[,-9])
  y <- prostate[,9]
  system.time(m<-regal(X, y, MaxIter=1000, d=10, NCores=8, verboseQ=TRUE))[3]
  ind <- rev(order(m[,6]))
  dotchart(m[ind,6], pch=19, cex=0.5, bg=rgb(1,1,0,0.4),
       color="blue", main="CPU times")

## End(Not run)
```

---

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

```
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

ESL. Hastie, Tibshirani and Friedman (2009). The Elements of Statistical Learning. 2nd Ed. Springer.

See Also

rxor, rdigitsBFOS, ShaoReg

Examples

```
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")
  })
```

---

rxor *Random XOR Samples*

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

```
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

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

ShaoReg *Synthetic Regression Data*

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

```
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 β is a vector of the regression coefficients of length p, X is the design matrix with n rows and p columns and ϵ is a vector of n independent normal random variables with mean zero and standard deviation sig. 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

Jun Shao (1993), Linear Model Selection by Cross-validation, Journal of the American Statistical Association, 88/422.

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

Nathalie Pochet, Frank De Smet, Johan A.K. Suykens and Bart L.R. De Moor (2004). Systematic benchmarking of microarray data classification: assessing the role of nonlinearity and dimensionality reduction. Bioinformatics Advance Access published July 1, 2004.

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

Nathalie Pochet, Frank De Smet, Johan A.K. Suykens and Bart L.R. De Moor (2004). Systematic benchmarking of microarray data classification: assessing the role of nonlinearity and dimensionality reduction. Bioinformatics Advance Access published July 1, 2004.

References

Terry Speed

See Also

`featureSelect`, `churnTrain`

Examples

```
yh_C50(SinghTrain, SinghTest)#0.235
dim(SinghTrain)
dim(SinghTest)
```

`smape` *Mean Absolute Percentage Error*

Description

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

Usage

```
smape(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

Examples

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

---

vifx *Variance Inflation Factor*

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])
```

yhat_CART *CART regression prediction*

Description

Regression prediction using tree method.

Usage

```
yhat_CART(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

```
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_CART(trdf, tedf)
```

yhat_gel *Elastic Net Regression Prediction*

Description

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

Usage

```
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

```
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

```
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

```
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

```
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

```
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)
```

yhat_nn *Nearest Neighbour Prediction*

Description

Nearest neighbour prediction

Usage

```
yhat_nn(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

```
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_nn(trdf, tedf)
```

yhat_plus *SCAD or MCP Regression Prediction*

Description

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

Usage

```
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

```
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

```
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_step *Backward Stagewise Regression with AIC or BIC*

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

```
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

```
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

```
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_SVM(trdf, tedf)
```

yh_C50 *C50 Prediction*

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
```

yh_CART *CART Prediction*

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_lda, yh_logistic, yh_qda

Examples

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

---

yh_kNN *kNN or NN prediction*

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

```
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

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

yh_logistic *Logistic Regression and Regularized Logistic Regression Prediction*

Description

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

Usage

```
yh_logistic(dfTr, dfTe, alpha = NULL)
```

Arguments

dfTr	Training data frame, last column factor response and other columns are numeric inputs.
dfTe	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 <- kyphosis[,c(2:4,1)]
set.seed(37771)
i <- sample(1:81, size=7, replace=TRUE)
dfTe <- z[i,]
i <- setdiff(1:81, i)
dfTr <- z[i,]
yh_logistic(dfTr, dfTe)
yh_logistic(dfTr, dfTe, alpha=1)
## Not run: #cross-validation, takes a few minutes
X <- kyphosis[,3:4]
y <- kyphosis[,4]
cgcv(X, y, yh=yh_logistic, NCores=8)
cgcv(X, y, yh=yh_logistic, NCores=8, alpha=1)
cgcv(X, y, yh=yh_logistic, NCores=8, alpha=0.5)
cgcv(X, y, yh=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

```
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

```
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

```
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.

Value

tba. Not fully implemented yet.

Author(s)

A. I. McLeod

Examples

```
require("MASS")
yh_qda(SinghTrain[,c(1:10, 101)], SinghTest[,c(1:10, 101)])#0.0588
```

---

yh_RF *Random Forest Prediction*

Description

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

Usage

```
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

```
yh_RF(SinghTrain, SinghTest)#0
```

yh_svm *Support Vector Machine Prediction*

Description

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

Usage

```
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

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

Index