Package 'Kira'

April 18, 2025

Type Package

Title Machine Learning

Version 1.0.7

Date 2025-04-19

Imports graphics, grDevices, MASS, stats

Depends R (>= 3.3.2)

Description Machine learning, containing several algorithms for supervised and unsupervised classification, in addition to a function that plots the Receiver Operating Characteristic (ROC) and Precision-Recall (PRC) curve graphs, and also a function that returns several metrics used for model evaluation, the latter can be used in ranking results from other packs.

License GPL-3

Encoding UTF-8

NeedsCompilation yes

Author Paulo Cesar Ossani [aut, cre] (<https://orcid.org/0000-0002-6617-8085>)

Maintainer Paulo Cesar Ossani <ossanipc@hotmail.com>

Repository CRAN

Date/Publication 2025-04-18 17:00:02 UTC

Contents

`Kira-package` *Machine learning and data mining.*

Description

Machine learning, containing several algorithms, in addition to functions that plot the graphs of the Receiver Operating Characteristic (ROC) and Precision-Recall (PRC) curve, and also a function that returns several metrics used to evaluate the models, the latter can be used in the classification results of other packages.

Details

Package:	Kira
Type:	Package
Version:	1.0.7
Date:	2025-04-19
License:	GPL(>= 3)
LazyLoad:	yes

This package contains:

- Algorithms for supervised classification: knn, linear (lda) and quadratic (qda) discriminant analysis, linear regression, etc.
- Algorithms for unsupervised classification: hierarchical, kmeans, etc.
- A function that plots the ROC and PRC curve.
- A function that returns a series of metrics from models.
- Functions that determine the ideal number of clusters: elbow and silhouette.

Author(s)

Paulo Cesar Ossani <ossanipc@hotmail.com>

References

Aha, D. W.; Kibler, D. and Albert, M. K. Instance-based learning algorithms. *Machine learning.* v.6, n.1, p.37-66. 1991.

Anitha, S.; Metilda, M. A. R. Y. An extensive investigation of outlier detection by cluster validation indices. *Ciencia e Tecnica Vitivinicola - A Science and Technology Journal*, v. 34, n. 2, p. 22-32, 2019. doi: 10.13140/RG.2.2.26801.63848

Charnet, R. at al. *Analise de modelos de regressao lienar,* 2a ed. Campinas: Editora da Unicamp, 2008. 357 p.

Chicco, D.; Warrens, M. J. and Jurman, G. The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. *IEEE Access, IEEE*, v. 9, p. 78368-78381, 2021.

Erich, S. Stop using the Elbow criterion for k-means and how to choose the number of clusters instead. *ACM SIGKDD Explorations Newsletter.* 25 (1): 36-42. arXiv:2212.12189. 2023. doi: 10.1145/3606274.3606278

Ferreira, D. F. *Estatistica Multivariada.* 2a ed. revisada e ampliada. Lavras: Editora UFLA, 2011. 676 p.

Kaufman, L. and Rousseeuw, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*, New York: John Wiley & Sons. 1990.

Martinez, W. L.; Martinez, A. R.; Solka, J. *Exploratory data analysis with MATLAB*. 2nd ed. New York: Chapman & Hall/CRC, 2010. 499 p.

Mingoti, S. A. *analysis de dados atraves de metodos de estatistica multivariada:* uma abordagem aplicada. Belo Horizonte: UFMG, 2005. 297 p.

Nicoletti, M. do C. O modelo de aprendizado de maquina baseado em exemplares: principais caracteristicas e algoritmos. Sao Carlos: EdUFSCar, 2005. 61 p.

Onumanyi, A. J.; Molokomme, D. N.; Isaac, S. J. and Abu-Mahfouz, A. M. Autoelbow: An automatic elbow detection method for estimating the number of clusters in a dataset. *Applied Sciences 12*, 15. 2022. doi: 10.3390/app12157515

Rencher, A. C. *Methods of multivariate analysis.* 2th. ed. New York: J.Wiley, 2002. 708 p.

Rencher, A. C. and Schaalje, G. B. *Linear models in statisctic.* 2th. ed. New Jersey: John & Sons, 2008. 672 p.

Rousseeuw P. J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53-65. 1987. doi: 10.1016/0377-0427(87)90125-7

Sugar, C. A. and James, G. M. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98, 463, 750-763. 2003. doi: 10.1198/016214503000000666

Venables, W. N. and Ripley, B. D. *Modern Applied Statistics with S.* Fourth edition. Springer, 2002.

Zhang, Y.; Mandziuk, J.; Quek, H. C. and Goh, W. Curvature-based method for determining the number of clusters. *Inf. Sci.* 415, 414-428, 2017. doi: 10.1016/j.ins.2017.05.024

`brute.force` *Brute force method for variable selection.*

Description

Brute force method used to determine the smallest number of variables in a supervised classification model.

Usage

```
brute.force(func = NA, train, test, class.train,
            class.test,  args = NA, measure = "Rate Hits",
            output = 10)
```

Arguments

func	Supervised classification function to be analyzed.
train	Data set of training, without classes.
test	Test data set.
class.train	Vector with training data class names.
class.test	Vector with test data class names.
args	Argument using in the classifier giving in 'func'.
measure	Measure to evaluate the model: "Rate Hits" (default), "Kappa Index", "Sensitivity", "Specificity", "Precision", "FP Rate", "FN Rate", "Negative Predictive Rate", "F-Score", "MCC", "ROC Are" or "PRC Area". If measure = NA returns all metrics ordered by 'Rate Hits'.
output	Number of elements with the best combinations of variables in the matrix 'best.model' (default = 10).

Value

best.model	Matrix with the names of the best combinations of variables, according to the evaluation measure used: accuracy, precision, recall etc.
text.model	Structure of the classification model used.

Author(s)

Paulo Cesar Ossani

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####

r <- (ncol(data) - 1)
```

```
res <- brute.force(func = "knn", train = data.train[,1:r],
                   test = data.test[,1:r], class.train = class.train,
                   class.test = class.test, args = "k = 1, dist = 'EUC'",
                   measure = "Rate Hits", output = 20)
res$best.model
res$text.model

res <- brute.force(func = "regression", train = data.train[,1:r],
                   test = data.test[,1:r], class.train = class.train,
                   class.test = class.test, args = "intercept = TRUE",
                   measure = "Rate Hits", output = 20)
res$best.model
res$text.model

test_a <- as.integer(rownames(data.test)) # test data index
class  <- data[,c(r+1)] # classes names
res <- brute.force(func = "lda", train = data[,1:r], test = test_a,
                   class.train = class, class.test = class.test,
                   args = "type = 'test', method = 'mle'",
                   measure = "Rate Hits", output = 20)
res$best.model
res$text.model
```

---

elbow *Elbow method to determine the optimal number of clusters.*

Description

Generates the Elbow graph and returns the ideal number of clusters.

Usage

```
elbow(data, k.max = 10, method = "AutoElbow", plot = TRUE,
      cut = TRUE, title = NA, xlabel = NA, ylabel = NA, size = 1.1,
      grid = TRUE, color = TRUE, savptc = FALSE, width = 3236,
      height = 2000, res = 300, casc = TRUE)
```

Arguments

data	Data with x and y coordinates.
k.max	Maximum number of clusters for comparison (default = 10).
method	Method used to find the ideal number k of clusters: "jump", "curvature", "Exp", "AutoElbow" (default).
plot	Indicates whether to plot the elbow graph (default = TRUE).
cut	Indicates whether to plot the best cluster indicative line (default = TRUE).
title	Title of the graphic, if not set, assumes the default text.
xlabel	Names the X axis, if not set, assumes the default text.

ylabel	Names the Y axis, if not set, assumes the default text.
size	Size of points on the graph and line thickness (default = 1.1).
grid	Put grid on graph (default = TRUE).
color	Colored graphic (default = TRUE).
savptc	Saves the graph image to a file (default = FALSE).
width	Graphic image width when savptc = TRUE (defaul = 3236).
height	Graphic image height when savptc = TRUE (default = 2000).
res	Nominal resolution in ppi of the graphic image when savptc = TRUE (default = 300).
casc	Cascade effect in the presentation of the graphic (default = TRUE).

Value

k.ideal	Ideal number of clusters.

Author(s)

Paulo Cesar Ossani

References

Erich, S. Stop using the Elbow criterion for k-means and how to choose the number of clusters instead. *ACM SIGKDD Explorations Newsletter.* 25 (1): 36-42. arXiv:2212.12189. 2023. doi: 10.1145/3606274.3606278

Sugar, C. A. and James, G. M. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98, 463, 750-763. 2003. doi: 10.1198/016214503000000666

Zhang, Y.; Mandziuk, J.; Quek, H. C. and Goh, W. Curvature-based method for determining the number of clusters. *Inf. Sci.* 415, 414-428, 2017. doi: 10.1016/j.ins.2017.05.024

Onumanyi, A. J.; Molokomme, D. N.; Isaac, S. J. and Abu-Mahfouz, A. M. Autoelbow: An automatic elbow detection method for estimating the number of clusters in a dataset. *Applied Sciences 12*, 15. 2022. doi: 10.3390/app12157515

Examples

```
data(iris) # data set

res <- elbow(data = iris[,1:4], k.max = 20, method = "AutoElbow", cut = TRUE,
             plot = TRUE, title = NA, xlabel = NA, ylabel = NA, size = 1.1,
             grid = TRUE, savptc = FALSE, width = 3236, color = TRUE,
             height = 2000, res = 300, casc = FALSE)

res$k.ideal # number of clusters
```

`hierarchical` *Hierarchical unsupervised classification.*

Description

Performs hierarchical unsupervised classification analysis in a data set.

Usage

```
hierarchical(data, titles = NA, analysis = "Obs", cor.abs = FALSE,
         normalize = FALSE, distance = "euclidean", method = "complete",
         horizontal = FALSE, num.groups = 0, lambda = 2, savptc = FALSE,
         width = 3236, height = 2000, res = 300, casc = TRUE)
```

Arguments

`data`	Data to be analyzed.
`titles`	Titles of the graphics, if not set, assumes the default text.
`analysis`	"Obs" for analysis on observations (default), "Var" for analysis on variables.
`cor.abs`	Matrix of absolute correlation case 'analysis' = "Var" (default = FALSE).
`normalize`	Normalize the data only for case 'analysis' = "Obs" (default = FALSE).
`distance`	Metric of the distances in case of hierarchical groupings: "euclidean" (default), "maximum", "manhattan", "canberra", "binary" or "minkowski". Case Analysis = "Var" the metric will be the correlation matrix, according to cor.abs.
`method`	Method for analyzing hierarchical groupings: "complete" (default), "ward.D", "ward.D2", "single", "average", "mcquitty", "median" or "centroid".
`horizontal`	Horizontal dendrogram (default = FALSE).
`num.groups`	Number of groups to be formed.
`lambda`	Value used in the minkowski distance.
`savptc`	Saves graphics images to files (default = FALSE).
`width`	Graphics images width when savptc = TRUE (defaul = 3236).
`height`	Graphics images height when savptc = TRUE (default = 2000).
`res`	Nominal resolution in ppi of the graphics images when savptc = TRUE (default = 300).
`casc`	Cascade effect in the presentation of the graphics (default = TRUE).

Value

Several graphics.

`tab.res`	Table with similarities and distances of the groups formed.
`groups`	Original data with groups formed.

res.groups	Results of the groups formed.
R.sqt	Result of the R squared.
sum.sqt	Total sum of squares.
mtx.dist	Matrix of the distances.

Author(s)

Paulo Cesar Ossani

References

Rencher, A. C. *Methods of multivariate analysis.* 2th. ed. New York: J.Wiley, 2002. 708 p.

Mingoti, S. A. *analysis de dados atraves de metodos de estatistica multivariada:* uma abordagem aplicada. Belo Horizonte: UFMG, 2005. 297 p.

Ferreira, D. F. *Estatistica Multivariada.* 2a ed. revisada e ampliada. Lavras: Editora UFLA, 2011. 676 p.

Examples

```
data(iris) # data set

data <- iris

res <- hierarchical(data[,1:4], titles = NA, analysis = "Obs", cor.abs = FALSE,
            normalize = FALSE, distance = "euclidean", method = "ward.D",
            horizontal = FALSE, num.groups = 3, savptc = FALSE, width = 3236,
            height = 2000, res = 300, casc = FALSE)

message("R squared: ", res$R.sqt)
# message("Total sum of squares: ", res$sum.sqt)
message("Groups formed: "); res$groups
# message("Table with similarities and distances:"); res$tab.res
# message("Table with the results of the groups:"); res$res.groups
# message("Distance Matrix:"); res$mtx.dist

#write.table(file=file.path(tempdir(),"GroupData.csv"), res$groups, sep=";",
#            dec=",",row.names = TRUE)
```

---

kmeans *kmeans unsupervised classification.*

Description

Performs kmeans unsupervised classification analysis in a data set.

Usage

```
kmeans(data, normalize = FALSE, num.groups = 2)
```

Arguments

data	Data to be analyzed.
normalize	Normalize the data (default = FALSE).
num.groups	Number of groups to be formed (default = 2).

Value

groups	Original data with groups formed.
res.groups	Results of the groups formed.
R.sqt	Result of the R squared.
sum.sqt	Total sum of squares.

Author(s)

Paulo Cesar Ossani

References

Rencher, A. C. *Methods of multivariate analysis.* 2th. ed. New York: J.Wiley, 2002. 708 p.

Mingoti, S. A. *analysis de dados atraves de metodos de estatistica multivariada:* uma abordagem aplicada. Belo Horizonte: UFMG, 2005. 297 p.

Ferreira, D. F. *Estatistica Multivariada.* 2a ed. revisada e ampliada. Lavras: Editora UFLA, 2011. 676 p.

Examples

```
data(iris) # data set

data <- iris

res <- kmeans(data[,1:4], normalize = FALSE, num.groups = 3)

message("R squared: ", res$R.sqt)
# message("Total sum of squares: ", res$sum.sqt)
message("Groups formed:"); res$groups
# message("Table with the results of the groups:"); res$res.groups

#write.table(file=file.path(tempdir(),"GroupData.csv"), res$groups, sep=";",
#            dec=",",row.names = TRUE)
```

knn *k-nearest neighbor (kNN) supervised classification method*

Description

Performs the k-nearest neighbor (kNN) supervised classification method.

Usage

```
knn(train, test, class, k = 1, dist = "euclidean", lambda = 3)
```

Arguments

`train`	Data set of training, without classes.
`test`	Test data set.
`class`	Vector with data classes names.
`k`	Number of nearest neighbors (default = 1).
`dist`	Distances used in the method: "euclidean" (default), "manhattan", "minkowski", "canberra", "maximum" or "chebyshev".
`lambda`	Value used in the minkowski distance (default = 3).

Value

`predict`	The classified factors of the test set.

Author(s)

Paulo Cesar Ossani

References

Aha, D. W.; Kibler, D. and Albert, M. K. Instance-based learning algorithms. *Machine learning.* v.6, n.1, p.37-66. 1991.

Nicoletti, M. do C. O modelo de aprendizado de maquina baseado em exemplares: principais caracteristicas e algoritmos. Sao Carlos: EdUFSCar, 2005. 61 p.

See Also

`plot_curve` and `results`

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####


dist = "euclidean"
# dist = "manhattan"
# dist = "minkowski"
# dist = "canberra"
# dist = "maximum"
# dist = "chebyshev"

k = 1
lambda = 5

r <- (ncol(data) - 1)
res <- knn(train = data.train[,1:r], test = data.test[,1:r], class = class.train,
           k = 1, dist = dist, lambda = lambda)

resp <- results(orig.class = class.test, predict = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix:"); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
message("Kappa coefficient: ", resp$kappa)
message("General results of the classes:"); resp$res.class
```

`lda` *Linear discriminant analysis (LDA).*

Description

Perform linear discriminant analysis.

Usage

```
lda(data, test = NA, class = NA, type = "train",
   method = "moment", prior = NA)
```

Arguments

data	Data to be classified.
test	Vector with indices that will be used in 'data' as test. For type = "train", one has test = NA.
class	Vector with data classes names.
type	Type of type: "train" - data training (default), or "test" - classifies the data of the vector "test".
method	Classification method: "mle" to MLEs, "mve" to use cov.mv, "moment" (default) for standard mean and variance estimators, or "t" for robust estimates based on the t distribution.
prior	Probabilities of occurrence of classes. If not specified, it will take the proportions of the classes. If specified, probabilities must follow the order of factor levels.

Value

predict	The classified factors of the set.

Author(s)

Paulo Cesar Ossani

References

Rencher, A. C. *Methods of multivariate analysis.* 2th. ed. New York: J.Wiley, 2002. 708 p.

Venables, W. N. and Ripley, B. D. *Modern Applied Statistics with S.* Fourth edition. Springer, 2002.

Mingoti, S. A. *Analise de dados atraves de metodos de estatistica multivariada:* uma abordagem aplicada. Belo Horizonte: UFMG, 2005. 297 p.

Ferreira, D. F. *Estatistica Multivariada.* 2a ed. revisada e ampliada. Lavras: Editora UFLA, 2011. 676 p.

See Also

plot_curve and results

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####

r <- (ncol(data) - 1)
class <- data[,c(r+1)] # classes names

## Data training example
res <- lda(data = data[,1:r], test = NA, class = class,
           type = "train", method = "moment", prior = NA)

resp <- results(orig.class = class, predict = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix:"); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
message("Kappa coefficient: ", resp$kappa)
message("General results of the classes:"); resp$res.class


## Data test example
class.table <- table(class) # table with the number of elements per class
prior <- as.double(class.table/sum(class.table))
test = as.integer(rownames(data.test)) # test data index

res <- lda(data = data[,1:r], test = test, class = class,
           type = "test", method = "mle", prior = prior)

resp <- results(orig.class = class.test, predict = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix: "); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
```

```
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
message("Kappa coefficient: ", resp$kappa)
message("General results of the classes:"); resp$res.class
```

---

plot_curve *Graphics of the results of the classification process*

Description

Return graphics of the results of the classification process.

Usage

```
plot_curve(data, type = "ROC", title = NA, xlabel = NA, ylabel = NA,
           posleg = 3, boxleg = FALSE, axis = TRUE, size = 1.1, grid = TRUE,
           color = TRUE, classcolor = NA, savptc = FALSE, width = 3236,
           height = 2000, res = 300, casc = TRUE)
```

Arguments

data	Data with x and y coordinates.
type	ROC (default) or PRC graphics type.
title	Title of the graphic, if not set, assumes the default text.
xlabel	Names the X axis, if not set, assumes the default text.
ylabel	Names the Y axis, if not set, assumes the default text.
posleg	0 with no caption, 1 for caption in the left upper corner, 2 for caption in the right upper corner, 3 for caption in the right lower corner (default), 4 for caption in the left lower corner.
boxleg	Puts the frame in the caption (default = TRUE).
axis	Put the diagonal axis on the graph (default = TRUE).
size	Size of the points in the graphs (default = 1.1).
grid	Put grid on graphs (default = TRUE).
color	Colored graphics (default = TRUE).
classcolor	Vector with the colors of the classes.
savptc	Saves graphics images to files (default = FALSE).
width	Graphics images width when savptc = TRUE (defaul = 3236).
height	Graphics images height when savptc = TRUE (default = 2000).
res	Nominal resolution in ppi of the graphics images when savptc = TRUE (default = 300).
casc	Cascade effect in the presentation of the graphic (default = TRUE).

Value

ROC or PRC curve.

Author(s)

Paulo Cesar Ossani

See Also

results

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####


dist = "euclidean"
# dist = "manhattan"
# dist = "minkowski"
# dist = "canberra"
# dist = "maximum"
# dist = "chebyshev"

k = 1
lambda = 5

r <- (ncol(data) - 1)
res <- knn(train = data.train[,1:r], test = data.test[,1:r], class = class.train,
           k = 1, dist = dist, lambda = lambda)

resp <- results(orig.class = class.test, predict = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix:"); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
```

```
message("Kappa coefficient: ", resp$kappa)
# message("Data for the ROC curve in classes:"); resp$roc.curve
# message("Data for the PRC curve in classes:"); resp$prc.curve
message("General results of the classes:"); resp$res.class

dat <- resp$roc.curve; tp = "roc"; ps = 3
# dat <- resp$prc.curve; tp = "prc"; ps = 4

plot_curve(data = dat, type = tp, title = NA, xlabel = NA, ylabel = NA,
           posleg = ps, boxleg = FALSE, axis = TRUE, size = 1.1, grid = TRUE,
           color = TRUE, classcolor = NA, savptc = FALSE,
           width = 3236, height = 2000, res = 300, casc = FALSE)
```

---

qda *Quadratic discriminant analysis (QDA).*

Description

Perform quadratic discriminant analysis.

Usage

```
qda(data, test = NA, class = NA, type = "train",
   method = "moment", prior = NA)
```

Arguments

data	Data to be classified.
test	Vector with indices that will be used in 'data' as test. For type = "train", one has test = NA.
class	Vector with data classes names.
type	Type of type: "train" - data training (default), or "test" - classifies the data of the vector "test".
method	Classification method: "mle" to MLEs, "mve" to use cov.mv, "moment" (default) for standard mean and variance estimators, or "t" for robust estimates based on the t distribution.
prior	Probabilities of occurrence of classes. If not specified, it will take the proportions of the classes. If specified, probabilities must follow the order of factor levels.

Value

predict	The classified factors of the set.

Author(s)

Paulo Cesar Ossani

References

Rencher, A. C. *Methods of multivariate analysis.* 2th. ed. New York: J.Wiley, 2002. 708 p.

Venables, W. N. and Ripley, B. D. *Modern Applied Statistics with S.* Fourth edition. Springer, 2002.

Mingoti, S. A. *Analise de dados atraves de metodos de estatistica multivariada:* uma abordagem aplicada. Belo Horizonte: UFMG, 2005. 297 p.

Ferreira, D. F. *Estatistica Multivariada.* 2a ed. revisada e ampliada. Lavras: Editora UFLA, 2011. 676 p.

See Also

plot_curve and results

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####

r <- (ncol(data) - 1)
class <- data[,c(r+1)] # classes names

## Data training example
res <- qda(data = data[,1:r], test = NA, class = class,
           type = "train", method = "moment", prior = NA)

resp <- results(orig.class = class, predict = res$predict)

message("Mean Squared Error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix: "); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
```

```
message("Kappa coefficient: ", resp$kappa)
message("General results of the classes:"); resp$res.class


## Data test example
class.table <- table(class) # table with the number of elements per class
prior <- as.double(class.table/sum(class.table))
test = as.integer(rownames(data.test)) # test data index

res <- qda(data = data[,1:r], test = test, class = class,
           type = "test", method = "mle", prior = prior)

resp <- results(orig.class = class.test, predic = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix: "); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
message("Kappa coefficient: ", resp$kappa)
message("General results of the classes:"); resp$res.class
```

`regression` *Linear regression supervised classification method*

Description

Performs supervised classification using the linear regression method.

Usage

```
regression(train, test, class, intercept = TRUE)
```

Arguments

`train`	Data set of training, without classes.
`test`	Test data set.
`class`	Vector with data classes names.
`intercept`	Consider the intercept in the regression (default = TRUE).

Value

`predict`	The classified factors of the test set.

Author(s)

Paulo Cesar Ossani

References

Charnet, R. at al. *Analise de modelos de regressao lienar,* 2a ed. Campinas: Editora da Unicamp, 2008. 357 p.

Rencher, A. C. and Schaalje, G. B. *Linear models in statisctic.* 2th. ed. New Jersey: John & Sons, 2008. 672 p.

Rencher, A. C. *Methods of multivariate analysis.* 2th. ed. New York: J.Wiley, 2002. 708 p.

See Also

plot_curve and results

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####

r <- (ncol(data) - 1)
res <- regression(train = data.train[,1:r], test = data.test[,1:r],
                  class = class.train, intercept = TRUE)

resp <- results(orig.class = class.test, predict = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix:"); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
message("Kappa coefficient: ", resp$kappa)
message("General results of the classes:"); resp$res.class
```

`results` *Results of the classification process*

Description

Returns the results of the classification process.

Usage

```
results(orig.class, predict)
```

Arguments

`orig.class`	Data with the original classes.
`predict`	Data with classes of results of classifiers.

Value

`mse`	Mean squared error.
`mae`	Mean absolute error.
`rae`	Relative absolute error.
`conf.mtx`	Confusion matrix.
`rate.hits`	Hit rate.
`rate.error`	Error rate.
`num.hits`	Number of correct instances.
`num.error`	Number of wrong instances.
`kappa`	Kappa coefficient.
`roc.curve`	Data for the ROC curve in classes.
`prc.curve`	Data for the PRC curve in classes.
`res.class`	General results of the classes: Sensitivity, Specificity, Precision, TP Rate, FP Rate, NP Rate, F-Score, MCC, ROC Area, PRC Area.

Author(s)

Paulo Cesar Ossani

References

Chicco, D.; Warrens, M. J. and Jurman, G. The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. *IEEE Access, IEEE*, v. 9, p. 78368-78381, 2021.

See Also

`plot_curve`

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####


dist = "euclidean"
# dist = "manhattan"
# dist = "minkowski"
# dist = "canberra"
# dist = "maximum"
# dist = "chebyshev"

k = 1
lambda = 5

r <- (ncol(data) - 1)
res <- knn(train = data.train[,1:r], test = data.test[,1:r], class = class.train,
           k = 1, dist = dist, lambda = lambda)

resp <- results(orig.class = class.test, predict = res$predict)

message("Mean squared error:"); resp$mse
message("Mean absolute error:"); resp$mae
message("Relative absolute error:"); resp$rae
message("Confusion matrix:"); resp$conf.mtx
message("Hit rate: ", resp$rate.hits)
message("Error rate: ", resp$rate.error)
message("Number of correct instances: ", resp$num.hits)
message("Number of wrong instances: ", resp$num.error)
message("Kappa coefficient: ", resp$kappa)
# message("Data for the ROC curve in classes:"); resp$roc.curve
# message("Data for the PRC curve in classes:"); resp$prc.curve
message("General results of the classes:"); resp$res.class

dat <- resp$roc.curve; tp = "roc"; ps = 3
# dat <- resp$prc.curve; tp = "prc"; ps = 4

plot_curve(data = dat, type = tp, title = NA, xlabel = NA, ylabel = NA,
           posleg = ps, boxleg = FALSE, axis = TRUE, size = 1.1, grid = TRUE,
           color = TRUE, classcolor = NA, savptc = FALSE, width = 3236,
           height = 2000, res = 300, casc = FALSE)
```

`silhouette` *Silhouette method to determine the optimal number of clusters.*

Description

Generates the silhouette graph and returns the ideal number of clusters in the k-means method.

Usage

```
silhouette(data, k.cluster = 2:10, plot = TRUE, cut = TRUE,
           title = NA, xlabel = NA, ylabel = NA, size = 1.1, grid = TRUE,
           color = TRUE, savptc = FALSE, width = 3236, height = 2000,
           res = 300, casc = TRUE)
```

Arguments

`data`	Data with x and y coordinates.
`k.cluster`	Cluster numbers for comparison in the k-means method (default = 2:10).
`plot`	Indicates whether to plot the silhouette graph (default = TRUE).
`cut`	Indicates whether to plot the best cluster indicative line (default = TRUE).
`title`	Title of the graphic, if not set, assumes the default text.
`xlabel`	Names the X axis, if not set, assumes the default text.
`ylabel`	Names the Y axis, if not set, assumes the default text.
`size`	Size of points on the graph and line thickness (default = 1.1).
`grid`	Put grid on graph (default = TRUE).
`color`	Colored graphic (default = TRUE).
`savptc`	Saves the graph image to a file (default = FALSE).
`width`	Graphic image width when savptc = TRUE (defaul = 3236).
`height`	Graphic image height when savptc = TRUE (default = 2000).
`res`	Nominal resolution in ppi of the graphic image when savptc = TRUE (default = 300).
`casc`	Cascade effect in the presentation of the graphic (default = TRUE).

Value

`k.ideal`	Ideal number of clusters.
`eve.si`	Vector with averages of silhouette indices of cluster groups (si).

Author(s)

Paulo Cesar Ossani

References

Anitha, S.; Metilda, M. A. R. Y. An extensive investigation of outlier detection by cluster validation indices. *Ciencia e Tecnica Vitivinicola - A Science and Technology Journal*, v. 34, n. 2, p. 22-32, 2019. doi: 10.13140/RG.2.2.26801.63848

Kaufman, L. and Rousseeuw, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*, New York: John Wiley & Sons. 1990.

Martinez, W. L.; Martinez, A. R.; Solka, J. *Exploratory data analysis with MATLAB*. 2nd ed. New York: Chapman & Hall/CRC, 2010. 499 p.

Rousseeuw P. J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53-65. 1987. doi: 10.1016/0377-0427(87)90125-7

Examples

```
data(iris) # data set

res <- silhouette(data = iris[,1:4], k.cluster = 2:10, cut = TRUE,
                  plot = TRUE, title = NA, xlabel = NA, ylabel = NA,
                  size = 1.1, grid = TRUE, savptc = FALSE, width = 3236,
                  color = TRUE, height = 2000, res = 300, casc = TRUE)

res$k.ideal # number of clusters
res$eve.si  # vector with averages of si indices


res <- silhouette(data = iris[,1:4], k.cluster = 3, cut = TRUE,
                  plot = TRUE, title = NA, xlabel = NA, ylabel = NA,
                  size = 1.1, grid = TRUE, savptc = FALSE, width = 3236,
                  color = TRUE, height = 2000, res = 300, casc = TRUE)

res$k.ideal # number of clusters
res$eve.si  # vector with averages of si indices
```

---

vote *Performs the supervised classification vote method.*

Description

Performs the supervised classification voting method, using maximum agreement between classifiers.

Usage

```
vote(mtx.algtms = NA)
```

Arguments

mtx.algtms	Matrix with the results of the supervised classification algorithms to be analyzed.

Value

predict The classified factors of the test set.

Author(s)

Paulo Cesar Ossani

References

Kittler, J.; Hatef, M.; Duin, R. P. W. and Matas, J. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 20(3):226-239. 1998. doi: 10.1109/34.667881

See Also

results

Examples

```
data(iris) # data set

data  <- iris
names <- colnames(data)
colnames(data) <- c(names[1:4],"class")

#### Start - hold out validation method ####
dat.sample = sample(2, nrow(data), replace = TRUE, prob = c(0.7,0.3))
data.train = data[dat.sample == 1,] # training data set
data.test  = data[dat.sample == 2,] # test data set
class.train = as.factor(data.train$class) # class names of the training data set
class.test  = as.factor(data.test$class)  # class names of the test data set
#### End - hold out validation method ####

test  <- as.integer(rownames(data.test)) # test data index
r     <- (ncol(data)-1)
class <- data[,c(r+1)] # classes names

mod1 <- knn(train = data.train[,1:r], test = data.test[,1:r],
            class = class.train, k = 1, dist = 'EUC')
mod2 <- knn(train = data.train[,1:r], test = data.test[,1:r],
            class = class.train, k = 2, dist = 'EUC')
mod3 <- lda(data = data[,1:r], test = test, class = class,
            type = 'test', method = 'moment', prior = NA)
mod4 <- qda(data = data[,1:r], test = test, class = class,
            type = 'test', method = 'mle', prior = NA)
mod5 <- qda(data = data[,1:r], test = test, class = class,
            type = 'test', method = 'moment', prior = NA)
mod6 <- regression(train = data.train[,1:r], test = data.test[,1:r],
                   class = class.train, intercept = TRUE)

mod <- cbind(as.data.frame(mod1$predict), mod2$predict, mod3$predict,
             mod4$predict, mod5$predict, mod6$predict)
```

```
res <- vote(mtx.algtms = mod)

resp <- results(orig.class = class.test, predict = res$predict)

print("Confusion matrix:"); resp$conf.mtx
cat("Hit rate:", resp$rate.hits,
    "\nError rate:", resp$rate.error,
    "\nNumber of correct instances:", resp$num.hits,
    "\nNumber of wrong instances:", resp$num.error,
    "\nKappa coefficient:", resp$kappa)
print("General results of the classes:"); resp$res.class
```

Index