Package 'PredPsych'

July 23, 2019

Type Package

Title Predictive Approaches in Psychology

Version 0.4

Date 2019-07-23

Author Atesh Koul

Maintainer Atesh Koul <atesh.koul@gmail.com>

Description Recent years have seen an increased interest in novel methods
for analyzing quantitative data from experimental psychology. Currently, however, they lack an established and accessible software framework. Many existing implementations provide no guidelines,
consisting of small code snippets, or sets of packages. In addition, the use of existing packages often requires advanced programming experience. 'PredPsych' is a user-friendly toolbox based on machine learning predictive algorithms. It comprises of multiple functionalities for multivariate analyses of quantitative behavioral data based on machine learning models.

License GPL-3

LazyData TRUE

Depends R (>= 3.5.0)

Imports plyr, ggplot2, caret, rpart, e1071, mclust, MASS, party, randomForest, statmod

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

Repository CRAN

NeedsCompilation no

Date/Publication 2019-07-23 08:20:05 UTC

R topics documented:

`classifyFun` *Generic Classification Analyses*

Description

function for performing generic classification Analysis

Usage

```
classifyFun(Data, classCol, selectedCols, cvType, ntrainTestFolds,
  nTrainFolds, modelTrainFolds, nTuneFolds, tuneFolds, foldSep, cvFraction,
  ranges = NULL, tune = FALSE, cost = 1, gamma = 0.5,
  classifierName = "svm", genclassifier, silent = FALSE,
  extendedResults = FALSE, SetSeed = TRUE, NewData = NULL, ...)
```

Arguments

`Data`	(dataframe) dataframe of the data
`classCol`	(numeric or string) column number that contains the variable to be predicted
`selectedCols`	(optional) (numeric or string) all the columns of data that would be used either as predictor or as feature
`cvType`	(optional) (string) which type of cross-validation scheme to follow; One of the following values: • folds = (default) k-fold cross-validation • LOSO = Leave-one-subject-out cross-validation • holdout = holdout Crossvalidation. Only a portion of data (cvFraction) is used for training. • LOTO = Leave-one-trial out cross-validation.
`ntrainTestFolds`	(optional) (parameter for only k-fold cross-validation) No. of folds for training and testing dataset
`nTrainFolds`	(optional) (parameter for only k-fold cross-validation) No. of folds in which to further divide Training dataset
`modelTrainFolds`	= (optional) (parameter for only k-fold cross-validation) specific folds from the first train/test split (ntrainTestFolds) to use for training
`nTuneFolds`	(optional) (parameter for only k-fold cross-validation) No. of folds for Tuning

tuneFolds	(optional) (parameter for only k-fold cross-validation) specific folds from the above nTuneFolds to use for tuning
foldSep	(numeric) (parameter for only Leave-One_subject Out) mandatory column number for Leave-one-subject out cross-validation.
cvFraction	(optional) (numeric) Fraction of data to keep for training data
ranges	(optional) (list) ranges for tuning support vector machine
tune	(optional) (logical) whether tuning of svm parameters should be performed or not
cost	(optional) (numeric) regularization parameter of svm
gamma	(optional) (numeric) rbf kernel parameter
classifierName	(optional) (string) name of the classifier to be used
genclassifier	(optional) (function or string) a classifier function or a name (e.g. Classifier.svm)
silent	(optional) (logical) whether to print messages or not
extendedResults	(optional) (logical) Return extended results with model and other metrics
SetSeed	(optional) (logical) Whether to setseed or not. use SetSeed to seed the random number generator to get consistent results; set false only for permutation tests
NewData	(optional) (dataframe) New Data frame features for which the class membership is requested
...	(optional) additional arguments for the function

Details

This function implements Classification Analysis. Classification Analysis is a supervised machine learning approach that attempts to identify holistic patters in the data and assign to it classes (classification). Given a set of features, a classification analysis automatically learns intrinsic patterns in the data to be able to predict respective classes. If the data features are informative about the classes, a high classification score would be achieved.

Value

Depending upon `extendedResults`. `extendedResults` = FALSE outputs Test accuracy `accTest` of discrimination; `extendedResults` = TRUE outputs Test accuracy `accTest` of discrimination, `accTestRun` discrimination for each run in case of cvType as LOSO,LOTO or Folds `ConfMatrix` Confusion matrices and `classificationResults` list of the cross-validation results including the model and `ConfusionMatrixResults` Overall cross-validated confusion matrix results

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern Classification. Wiley-Interscience (Vol. 24).

Vapnik, V. (1995). The Nature of statistical Learning Theory. Springer-Verlag New York.

Hsu, C. C., Chang, C. C., & Lin, C. C. (2003). A practical guide to support vector classification, 1(1), 1-16.

Examples

```
# classification analysis with SVM
Results <- classifyFun(Data = KinData,classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),cvType="holdout")

# Output:

# Performing Classification Analysis
#
# Performing holdout Cross-validation
# genclassifier was not specified,
#   Using default value of Classifier.svm (genclassifier = Classifier.svm)"
#
# cvFraction was not specified,
#  Using default value of 0.8 (cvFraction = 0.8)
#
# Proportion of Test/Train Data was :  0.2470588
# [1] "Test holdout Accuracy is  0.65"
# holdout classification Analysis:
# cvFraction : 0.8
# Test Accuracy 0.65
# *Legend:
# cvFraction = Fraction of data to keep for training data
# Test Accuracy = Accuracy from the Testing dataset

# Alternate uses:
# perform a k-folds cross-validated classification analysis:
Results <- classifyFun(Data = KinData,classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),cvType = "folds")

# use extendedResults as well as tuning
Results <- classifyFun(Data = KinData,classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
cvType = "folds",extendedResults = TRUE,tune=TRUE)
```

`ClassPerm` *Permutation Analysis for classification*

Description

simple function to create permutation testing of a classifier

Usage

```
ClassPerm(Data, classCol, selectedCols, classifierFun, nSims = 1000,
  plot = TRUE, silent = FALSE, progress_bar = progress_time(), ...)
```

Arguments

`Data`	(dataframe) dataframe of the data
`classCol`	(numeric or string) column number that contains the variable to be predicted
`selectedCols`	(optional) (numeric or string) all the columns of data that would be used either as predictor or as feature
`classifierFun`	(optional) (function) classifier function
`nSims`	(optional) (numeric) number of simulations
`plot`	(optional) (logical) whether to plot null accuracy distribution
`silent`	(optional) (logical) whether to print messages or not
`progress_bar`	(optional) the type of progress bar to be utilized
`...`	(optional) additional arguments for the function

Details

The function implements Permutation tests for classification. Permutation tests are a set of non-parametric methods for hypothesis testing without assuming a particular distribution (Good, 2005). In case of classification analysis, this requires shuffling the labels of the dataset (i.e. randomly shuffling classes/conditions between observations) and calculating accuracies obtained.

Value

Returns `actualAcc` of the classification analysis, `p-value` from permutation testing, `nullAcc` distribution of the permutation `figure` containing null distribution

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Phipson, B., & Smyth, G. K. (2010). Permutation P-values Should Never Be Zero: Calculating Exact P-values When Permutations Are Randomly Drawn. Statistical Applications in Genetics and Molecular Biology, 9(1), 1544-6115.

Ojala, M. & Garriga, G. C. Permutation Tests for Studying Classifier Performance. J. Mach. Learn. Res. 11, 1833-1863 (2010).

Good, P. (2005). Permutation, Parametric and Bootstrap Tests of Hypotheses. New York: Springer-Verlag.

Examples

```
# perform a permutation testing for 10% of the kinematics movement data#'
# not run
# PermutationResult <- ClassPerm(Data = KinData, classCol = 1,
# selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112), nSims = 1000,cvType = "holdout")
# Output:
# Performing Permutation Analysis for Classification
#
# Performing Cross-validation
#
# Performing holdout Cross-validation
# genclassifier was not specified,
#  Using default value of Classifier.svm (genclassifier = Classifier.svm)
#
# cvFraction was not specified,
#  Using default value of 0.8 (cvFraction = 0.8)
#
# Proportion of Test/Train Data was :  0.2470588
# [1] "Test holdout Accuracy is  0.65"
# holdout classification Analysis:
# cvFraction : 0.8
# Test Accuracy 0.65
# *Legend:
# cvFraction = Fraction of data to keep for training data
# Test Accuracy = Accuracy from the Testing dataset
#
# Performing permutation testing...
# Performing 1000 simulations
# |=========================================================
# ============================================================|100%
#                      Completed after 2 m
# The p-value of the permutation testing is 0.001
# p-value generated using the approximate method for p-value calculation.
# See Phipson, B. & Gordon K., S. (2010) for details


# Using LinearDA instead as function
# not run
# PermutationResult <- ClassPerm(Data = KinData, classCol = 1,
# selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112), nSims = 1000,classifierFun = Lin


# Any minimalistic function can be used
# The ClassPerm function sends the dataframe Data, classCol,
# selectedCols as arguments
# not run
# myMinimalFun <- function(...){
# ***Calculate Error function as you want***
# return(accTest)
# }
# Use the function for permutation testing e.g.
# Results <- ClassPerm(Data = KinData, classCol=1,
```

```
# selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
# nSims = 1000,classifierFun = myMinimalFun)
```

`DimensionRed` *Generic Dimensionallity Reduction Function*

Description

A simple function to perform dimensionality reduction

Usage

```
DimensionRed(Data, method = "MDS", selectedCols, outcome = NA,
  plot = FALSE, silent = FALSE, ...)
```

Arguments

`Data`	(dataframe) a data frame with variable/feature columns
`method`	(optional) (character) Dimensionality reduction method to be used
`selectedCols`	(optional)(numeric) which columns should be treated as data(features/columns) (defaults to all columns)
`outcome`	(optional)(vector) optional vector for visualising plots
`plot`	(optional)(logical) To plot or not to plot
`silent`	(optional) (logical) whether to print messages or not
`...`	(optional) additional arguments for the function

Details

Dimensionality Reduction is the process of reducing the dimensions of the dataset. Multivariate data, even though are useful in getting an overall understanding of the underlying phenomena, do not permit easy interpretability. Moreover, variables in such data often are correlated with each other .For these reasons, it might be imperative to reduce the dimensions of the data. Various models have been developed for such dimensionality reduction. Of these, MDS and PCA has been demonstrated in the current implementation.

Value

Data frame with `Results`

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. (M. Jordan, J. Kleinberg, & B. Scholkopf, Eds.) (1st ed.). Springer-Verlag New York.

Cox, T. F., & Cox, M. A. A. (2000). Multidimensional scaling (Second ed.). Chapman & Hall/CRC.

Examples

```
# reducing dimension of Grip aperture from 10 to 2
GripAperture <- DimensionRed(KinData,selectedCols = 12:21,
outcome = KinData[,"Object.Size"],plot = TRUE)
```

`DTModel` *Generic Decision Tree Function*

Description

A simple function to create Decision Trees

Usage

```
DTModel(Data, classCol, selectedCols, tree, cvType, nTrainFolds,
  ntrainTestFolds, modelTrainFolds, foldSep, cvFraction,
  extendedResults = FALSE, SetSeed = TRUE, silent = FALSE,
  NewData = NULL, ...)
```

Arguments

`Data`	(dataframe) a data frame with regressors and response
`classCol`	(numeric or string) which column should be used as response col
`selectedCols`	(optional) (numeric or string) which columns should be treated as data(features + response) (defaults to all columns)
`tree`	which decision tree model to implement; One of the following values: • CART = Classification And Regression Tree; • CARTNACV = Crossvalidated CART Tree removing missing values; • CARTCV = Crossvalidated CART Tree With missing values; • CF = Conditional inference framework Tree; • RF = Random Forest Tree;
`cvType`	(optional) (string) which type of cross-validation scheme to follow - only in case of CARTCV or CARTNACV; One of the following values: • folds = (default) k-fold cross-validation • LOSO = Leave-one-subject-out cross-validation • holdout = holdout Crossvalidation. Only a portion of data (cvFraction) is used for training.

- LOTO = Leave-one-trial out cross-validation.

`nTrainFolds` (optional) (parameter for only k-fold cross-validation) No. of folds in which to further divide Training dataset

`ntrainTestFolds` (optional) (parameter for only k-fold cross-validation) No. of folds for training and testing dataset

`modelTrainFolds` = (optional) (parameter for only k-fold cross-validation) specific folds from the first train/test split (ntrainTestFolds) to use for training

`foldSep` (numeric) (parameter for only Leave-One_subject Out) mandatory column number for Leave-one-subject out cross-validation.

`cvFraction` (optional) (numeric) Fraction of data to keep for training data

`extendedResults` (optional) (logical) Return extended results with model and other metrics

`SetSeed` (optional) (logical) Whether to setseed or not. use SetSeed to seed the random number generator to get consistent results;

`silent` (optional) (logical) whether to print messages or not

`NewData` (optional) (dataframe) New Data frame features for which the class membership is requested

`...` (optional) additional arguments for the function

Details

The function implements the Decision Tree models (DT models). DT models fall under the general "Tree based methods" involving generation of a recursive binary tree (Hastie et al., 2009). In terms of input, DT models can handle both continuous and categorical variables as well as missing data. From the input data, DT models build a set of logical "if ..then" rules that permit accurate prediction of the input cases.

The function "rpart" handles the missing data by creating surrogate variables instead of removing them entirely (Therneau, & Atkinson, 1997). This could be useful in case the data contains multiple missing values.

Unlike regression methods like GLMs, Decision Trees are more flexible and can model nonlinear interactions.

Value

model result for the input tree `Results` or Test accuracy `accTest` based on `tree`. If `extendedResults` = TRUE outputs Test accuracy `accTest` of discrimination,`ConfMatrix` Confusion matrices and `fit` the model and `ConfusionMatrixResults` Overall cross-validated confusion matrix results

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer Series in Statistics (2nd ed., Vol. 1). New York, NY: Springer New York.

Terry Therneau, Beth Atkinson and Brian Ripley (2015). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-10. https://CRAN.R-project.org/package=rpart

Therneau, T. M., & Atkinson, E. J. (1997). An introduction to recursive partitioning using the RPART routines (Vol. 61, p. 452). Mayo Foundation: Technical report.

Examples

```
# generate a cart model for 10% of the data with cross-validation
model <- DTModel(Data = KinData,classCol=1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112), tree='CARTCV',cvType = "holdout")
# Output:
# Performing Decision Tree Analysis
#
# [1] "Generating crossvalidated Tree With Missing Values"
#
# Performing holdout Cross-validation
#
# cvFraction was not specified,
#  Using default value of 0.8 (cvFraction = 0.8)"
# Proportion of Test/Train Data was :  0.2470588
#
# [1] "Test holdout Accuracy is  0.62"
# holdout CART Analysis:
# cvFraction : 0.8
# Test Accuracy 0.62
# *Legend:
# cvFraction = Fraction of data to keep for training data
# Test Accuracy = Accuracy from the Testing dataset

#' # --CART MOdel --

# Alternate uses:
# k-fold cross-validation with removing missing values
model <- DTModel(Data = KinData,classCol=1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
tree='CARTNACV',cvType="folds")

# holdout cross-validation without removing missing values
model <- DTModel(Data = KinData,classCol=1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
tree='CARTCV',cvType = "holdout")

# k-fold cross-validation without removing missing values
model <- DTModel(Data = KinData,classCol=1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
tree='CARTCV',cvType="folds")
```

`fscore` *f-score*

Description

A simple function to generate F-scores (Fisher scores) for ranking features

Usage

```
fscore(Data, classCol, featureCol, silent = FALSE)
```

Arguments

`Data`	(dataframe) Data dataframe
`classCol`	(numeric) column with different classes
`featureCol`	(numeric) all the columns that contain features
`silent`	(optional) (logical) whether to print messages or not

Details

The function implements F-score for feature selection. F-score provides a measure of how well a single feature at a time can discriminate between different classes. The higher the F-score, the better the discriminatory power of that feature

The F-score is calculated for two classes

Value

named numeric `f-scores`

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern Classification. Wiley-Interscience (Vol. 24).

Chen, Y., & Lin, C.-J. (2006). Combining SVMs with Various Feature Selection Strategies. In I. Guyon, M. Nikravesh, S. Gunn, & L. A. Zadeh (Eds.), Feature Extraction: Foundations and Applications (Vol. 324, pp. 315-324). Berlin, Heidelberg: Springer Berlin Heidelberg.

Examples

```
# calculate f-scores for 10% of movement
fscore(KinData,classCol = 1,featureCol = c(2,12,22,32,42,52,62,72,82,92,102,112))
# Output:
# Performing Feature selection f-score analysis
# --f-scores--
```

`KinData` *Kinematics Dataset A dataset containing part of the motion capture dataset freely available in the publication (Ansuini et al., 2015).The dataset was obtained by recording 15 naive participants performing reach-to-grasp movements towards two differently sized objects: a small object (i.e., hazelnut) and a large object (i.e., grapefruit). The variables are as follows:*

Description

- Object Size : Size of the to-be-grasped object (1 = small, 2 = large)
- Wrist_Velocity_01 .. Wrist_Height_10: module of the velocity of the wrist marker (mm/sec) from 10% (_01) to 100% (_10) of the movement
- Grip_Aperture_01 .. Grip_Aperture_10 Distance between the marker placed on thumb tip and that placed on the tip of the index finger (mm) from 10% (_01) to 100% (_10) of the movement
- Wrist_Height_01 .. Wrist_Height_10 z-component of the wrist marker (mm) from 10% (_01) to 100% (_10) of the movement
- x_index_01 .. x_index_10 : x-coordinates for the index with respect to F-local (mm) from 10% (_01) to 100% (_10) of the movement
- y_index_01 .. y_index_10 : y-coordinates for the index with respect to F-local (mm) from 10% (_01) to 100% (_10) of the movement
- z_index_01 .. z_index_10 : z-coordinates for the index with respect to F-local (mm) from 10% (_01) to 100% (_10) of the movement
- x_thumb_01 .. x_thumb_10 : x-coordinates for the thumb with respect to F-local (mm) from 10% (_01) to 100% (_10) of the movement
- y_thumb_01 .. y_thumb_10 : y-coordinates for the thumb with respect to F-local (mm) from 10% (_01) to 100% (_10) of the movement
- z_thumb_01 .. z_thumb_10 : z-coordinates for the thumb with respect to F-local (mm) from 10% (_01) to 100% (_10) of the movement
- x_finger_plane_01 .. x_finger_plane_10 x-components of the thumb-index plane from 10% (_01) to 100% (_10) of the movement
- y_finger_plane_01 .. y_finger_plane_10 y-components of the thumb-index plane from 10% (_01) to 100% (_10) of the movement
- z_finger_plane_01 .. z_finger_plane_10 z-components of the thumb-index plane from 10% (_01) to 100% (_10) of the movement

Usage

```
data(KinData)
```

Format

A data frame with 848 rows and 121 variables

LinearDA *Cross-validated Linear Discriminant Analysis*

Description

A simple function to perform cross-validated Linear Discriminant Analysis

Usage

```
LinearDA(Data, classCol, selectedCols, cvType, nTrainFolds,
  ntrainTestFolds, modelTrainFolds, foldSep, CV = FALSE, cvFraction,
  extendedResults = FALSE, SetSeed = TRUE, silent = FALSE,
  NewData = NULL, ...)
```

Arguments

`Data`	(dataframe) Data dataframe
`classCol`	(numeric or string) column number that contains the variable to be predicted
`selectedCols`	(optional) (numeric or string) all the columns of data that would be used either as predictor or as feature
`cvType`	(optional) (string) which type of cross-validation scheme to follow; One of the following values: • folds = (default) k-fold cross-validation • LOSO = Leave-one-subject-out cross-validation • holdout = holdout Crossvalidation. Only a portion of data (cvFraction) is used for training. • LOTO = Leave-one-trial out cross-validation.
`nTrainFolds`	= (optional) (parameter for only k-fold cross-validation) No. of folds in which to further divide Training dataset
`ntrainTestFolds`	= (optional) (parameter for only k-fold cross-validation) No. of folds for training and testing dataset
`modelTrainFolds`	= (optional) (parameter for only k-fold cross-validation) specific folds from the first train/test split (ntrainTestFolds) to use for training
`foldSep`	(numeric) (parameter for only Leave-One_subject Out) mandatory column number for Leave-one-subject out cross-validation.

CV	(optional) (logical) perform Cross validation of training dataset? If TRUE, posterior probabilites are present with the model
cvFraction	(optional) (numeric) Fraction of data to keep for training data
extendedResults	(optional) (logical) Return extended results with model and other metrics
SetSeed	(optional) (logical) Whether to setseed or not. use SetSeed to seed the random number generator to get consistent results; set false only for permutation tests
silent	(optional) (logical) whether to print messages or not
NewData	(optional) (dataframe) New Data frame features for which the class membership is requested
...	(optional) additional arguments for the function

Details

The function implements Linear Disciminant Analysis, a simple algorithm for classification based analyses .LDA builds a model composed of a number of discriminant functions based on linear combinations of data features that provide the best discrimination between two or more conditions/classes. The aim of the statistical analysis in LDA is thus to combine the data features scores in a way that a single new composite variable, the discriminant function, is produced (for details see Fisher, 1936; Rao, 1948)).

Value

Depending upon extendedResults. extendedResults = FALSE outputs Test accuracy accTest of discrimination; extendedResults = TRUE outputs Test accuracy accTest of discrimination, ConfusionMatrixResults Overall cross-validated confusion matrix results,ConfMatrix Confusion matrices and fitLDA the fit cross-validated LDA model. If CV = TRUE , Posterior probabilities are generated and stored in the model.

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, 7(2), 179-188.

Rao, C. (1948). The Utilization of Multiple Measurements in Problems of Biological Classification. In Journal of the Royal Statistical Society. Series B (Methodological) (Vol. 10, pp. 159-203).

Examples

```
# simple model with holdout data partition of 80% and no extended results
LDAModel <- LinearDA(Data = KinData, classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),cvType="holdout")
# Output:
#
```

```
# Performing Linear Discriminant Analysis
#
#
# Performing holdout Cross-validation
#
# cvFraction was not specified,
#  Using default value of 0.8 (80%) fraction for training (cvFraction = 0.8)
#
# Proportion of Test/Train Data was :  0.2470588
# Predicted
# Actual  1  2
# 1 51 32
# 2 40 45
# [1] "Test holdout Accuracy is 0.57"
# holdout LDA Analysis:
# cvFraction : 0.8
# Test Accuracy 0.57
# *Legend:
# cvFraction = Fraction of data to keep for training data
# Test Accuracy = mean accuracy from the Testing dataset

# alt uses:
# holdout cross-validation with 80% training data
LDAModel <- LinearDA(Data = KinData, classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
CV=FALSE,cvFraction = 0.8,extendedResults = TRUE,cvType="holdout")

# For a 10 fold cross-validation without outputting messages
LDAModel <-  LinearDA(Data = KinData, classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),
extendedResults = FALSE,cvType = "folds",nTrainFolds=10,silent = TRUE)
```

`ModelCluster` *Model based Clustering*

Description

A simple function to perform Model based cluster Analysis :

Usage

```
ModelCluster(Data, NewData = NULL, G, silent = FALSE, ...)
```

Arguments

`Data`	(dataframe) Data dataframe
`NewData`	(optional) (dataframe) New Data frame for which the class membership is requested

G	(optional) (numeric) No. of components to verify
silent	(optional) (logical) whether to print messages or not
...	(optional) additional arguments for the function

Details

The function implements Model based clustering in predictive framework. Model based clustering approaches provide a structured way of choosing number of clusters (C. Fraley & Raftery, 1998). Data are considered to be generated from a set of Gaussian distributions (components or clusters) i.e. as a mixture of these components (mixture models). Instead of using heuristics, model based clustering approximates Bayes factor (utilizing Bayesian information Criterion) to determine the model with the highest evidence (as provided by the data).

Value

`class membership` of the clustered `NewData`

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Han, J., Kamber, M., & Pei, J. (2012). Cluster Analysis. In Data Mining (pp. 443-495). Elsevier.

Fraley, C., & Raftery, a E. (1998). How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. The Computer Journal, 41(8), 578-588.

Examples

```
# clustering kinematics data at 10% of movement
# not run
# cluster_time <- ModelCluster(KinData[,c(2,12,22,32,42,52,62,72,82,92,102,112)],G=1:12)
# Output:
# Performing Cluster analysis
# --cluster Results --
```

---

overallConfusionMetrics

Confusion Matrix metrics for Cross-validation

Description

A simple function to generate confusion matrix metrics for cross-validated Analyses

Usage

```
overallConfusionMetrics(confusionMat)
```

Arguments

`confusionMat` (confusion matrix or a list of it) Input confusion Matrix generated by function confusionMatrix from caret library

Details

A function to output confusion matrices and related metrics of sensitivity, specificity, precision, recall and other metrics for cross-validated analyses. There are multiple documented ways of calculating confusion matrix metrics for cross-validation (see Forman and Scholz 2012 for details on F1 score). The current procedure generates a final bigger confusion matrix and calculates the measures of sensitivity, specificity etc. on this matrix (instead of averaging sensitivities, specificities in each fold).

The intuition from (Kelleher, Namee and D'Arcy 2015) is:

"When we have a small dataset (introducing the possibility of a lucky split) measuring aggregate performance using a set of models gives a better estimate of post-deployment performance than measuring performance using a single model."

In addition, (Forman and Scholz 2010) using simulation studies show that F1 values calculated this way are less biased.

Value

A list with metrics as generated by confusionMatrix function in caret library.

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia

<atesh.koul@gmail.com>

References

Kelleher, J. D., Namee, B. Mac & D'Arcy, A. Fundamentals of Machine Learning for Predictive Data Analytics. (The MIT Press, 2015). Section 8.4.1.2 Elkan, C. Evaluating Classifiers. (2012).https://pdfs.semanticscholar.org/2bdc/61752a02783aa0e69e92fe6f9b449916a095.pdf pp. 4

Forman, G. & Scholz, M. Apples-to-apples in cross-validation studies. ACM SIGKDD Explor. Newsl. 12, 49 (2010).

Examples

```
# Result from a confusion matrix
confusionMat <- list(table = matrix(c(110,29,80,531),ncol = 2,
dimnames = list(Prediction = c(1,2),Reference = c(1,2))))
overallConfusionMetrics(confusionMat)

# Output:
#
# Confusion Matrix and Statistics
#           Reference
# Predicted   1   2
```

```
#          1 110  80
#          2  29 531
# Accuracy : 0.8547
# 95% CI : (0.8274, 0.8791)
# No Information Rate : 0.8147
# P-Value [Acc > NIR] : 0.002214
#
# Kappa : 0.5785
# Mcnemar's Test P-Value : 1.675e-06
#
# Sensitivity : 0.7914
# Specificity : 0.8691
# Pos Pred Value : 0.5789
# Neg Pred Value : 0.9482
# Prevalence : 0.1853
# Detection Rate : 0.1467
# Detection Prevalence : 0.2533
# Balanced Accuracy : 0.8302
#
# 'Positive' Class : 1

# Alternative (realistic) examples
Results <- classifyFun(Data = KinData,classCol = 1,
selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112),cvType = "folds",
extendedResults = TRUE)

overallConfusionMetrics(Results$ConfMatrix)
```

`predictNewData` *Predict Class membership for New Data*

Description

A simple function to predict class membership for new data

Usage

```
predictNewData(model, NewData, ...)
```

Arguments

`model`	(model) Classifier model obtained from a classification analysis
`NewData`	(optional) (dataframe) New Data frame features for which the class membership is requested
`...`	(optional) additional arguments for the function

Details

A function to generate predictions on a new dataset based on a previously estimated classifier model. This could be generated from LinearDA, classifyFun or DTMOdel functions.

Value

Predictions for each case in the NewData.

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia
<atesh.koul@gmail.com>

PredPsych *PredPsych.*

Description

PredPsych.

Details

"PredPsych" is a user-friendly, R toolbox based on machine learning predictive algorithms.

Author(s)

Atesh Koul, C'MON unit, Istituto Italiano di Tecnologia
<atesh.koul@gmail.com>

References

Koul, A., Becchio, C., & Cavallo, A. (2017, March 21). PredPsych: A toolbox for predictive machine learning based approach in experimental psychology research. Retrieved from osf.io/preprints/psyarxiv/pvjac