# Package ‘PredPsych’

December 14, 2016

**Type** Package

**Title** Generic Package for Predictive approaches in Psychology

**Version** 0.1

**Date** 2015-06-18

**Author** Atesh Koul

**Maintainer** <atesh.koul@iit.it>

**Description** The functions for Predictive approaches in Psychology

**License** GPL-V3

**LazyData** TRUE

**Depends** R (>= 3.1.0)

**Imports** plyr, ggplot2, openxlsx, caret, rpart, plotly, e1071, mclust

**RoxygenNote** 5.0.1

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

## R topics documented:

`classifyFun` *Generic Classification Analyses*

## Description

function for performing generic classification Analysis

## Usage

```
classifyFun(Data, classCol, selectedCols, ranges = NULL, tune = FALSE,
  cost = 1, gamma = 0.5, classifierName = "svm",
  genclassifier = Classifier.svm, silent = FALSE, SetSeed = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `Data` | (dataframe) dataframe of the data |
| `classCol` | (numeric) column number that contains the variable to be predicted |
| `selectedCols` | (optional) (numeric) all the columns of data that would be used either as predictor or as feature |
| `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 on mean accuracy or not |
| `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 |

## 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

Outputs Crossvalidation accuracy `acc` and Test accuracy `accTest`

## Author(s)

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

`<atesh.koul@iit.it>`

## 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))
# output
# [1] "Begining k-fold Classification"
# [1] "Mean CV Accuracy 0.66"
# [1] "Mean Test Accuracy 0.62"
```

---

`ClassPerm` *Permutation Analysis for classification*

---

## Description

simple function to create permutation testing of a classifier

## Usage

```
ClassPerm(Data, classCol, selectedCols, classifierFun, nSims = 1000, ...)
```

## Arguments

| | |
|---|---|
| `Data` | (dataframe) dataframe of the data |
| `classCol` | (numeric) column number that contains the variable to be predicted |
| `selectedCols` | (optional) (numeric) 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 |

## 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@iit.it>`

## Examples

```
# perform a permutation testing for 10% of the kinematics movement data
PermutationResult <- ClassPerm(Data = KinData, classCol = 1,
 selectedCols = c(1,2,12,22,32,42,52,62,72,82,92,102,112), nSims = 1000)
```

---

DimensionRed *Generic Dimensionallity Reduction Function*

---

## Description

A simple function to perform dimensionality reduction

## Usage

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

## Arguments

| | |
|---|---|
| Data | (dataframe) a data frame with variable/feature columns |
| 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 |

## 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@iit.it>

## 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, ...)
```

### Arguments

| | |
|---|---|
| Data | (dataframe) a data frame with regressors and response |
| classCol | (numeric) which column should be used as response col |
| selectedCols | (optional)(numeric) 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: <br>• CART = Classification And Regression Tree;<br>• CARTNAHF = Crossvalidated Half Model CART Tree removing missing values;<br>• CARTHF = Crossvalidated Half Model CART Tree With missing values;<br>• CF = Conditional inference framework Tree;<br>• RF = Random Forest Tree; |

### 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.

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

### Value

model result for the input tree `Results`

### Author(s)

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

<atesh.koul@iit.it>

### Examples

```
# generate a cart model for 10% of the data with cross-validation
model <- DTModel(Data = KinData[,c(1,2,12,22,32,42,52,62,72,82,92,102,112)],classCol=1,tree='CARTHF')
```

`fscore` *f-score*

## Description

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

## Usage

```
fscore(Data, classCol, featureCol)
```

## Arguments

| | |
|---|---|
| `Data` | (dataframe) Data dataframe |
| `classCol` | (numeric) column with different classes |
| `featureCol` | (numeric) all the columns that contain features |

## 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

## Value

named numeric `f-scores`

## Author(s)

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

`<atesh.koul@iit.it>`

## 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))
```

`LinearDA` *Cross-validated Linear Discriminant Analysis*

## Description

A simple function to perform cross-validated Linear Discriminant Analysis

## Usage

```
LinearDA(Data, classCol, selectedCols, CV = FALSE, cvFraction = 0.8,
  extendedResults = FALSE, SetSeed = TRUE, cvType = "createDataPartition",
  k = 10, ...)
```

## Arguments

| | |
|---|---|
| `Data` | (dataframe) Data dataframe |
| `classCol` | (numeric) column number that contains the variable to be predicted |
| `selectedCols` | (optional) (numeric) all the columns of data that would be used either as predictor or as feature |
| `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? |
| `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 |
| `cvType` | (optional) (string) type of cross validation to perform if cvType = 'createDataPartition' a portion of data (cvFraction) is used, For cvType = 'Folds', a n-fold cross validation is performed. |
| `k` | (optional) (numeric) the number of folds to use in case cvType = 'Folds' |

## 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 = `Acc` of discrimination () `extendedResults` TRUE `Acc` Accuracy of discrimination 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@iit.it>

## Examples

```
# simple model with 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))
#outout
#       Predicted
#Actual  1  2
#1 51 32
#2 40 45
#"The accuracy of discrimination was 0.57"

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)
```

ModelCluster *Model based Clustering*

## Description

A simple function to perform Model based cluster Analysis :

## Usage

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

## 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 |

## 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@iit.it>

## Examples

```
# clustering kinematics data at 10% of movement
cluster_time <- ModelCluster(KinData[,c(2,12,22,32,42,52,62,72,82,92,102,112)],G=1:12)
```

PredPsych *PredPsych.*

## Description

PredPsych.

## Details

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

# Index