Package ‘FuzzyClass’

May 27, 2022

Title Fuzzy and Non-Fuzzy Classifiers
Version 0.1.1
Description Provides classifiers that can be used for discrete variables and for continuous
variables based on the idea of Naive Bayes and Fuzzy Naive Bayes considering some statistical dis-
tributions of articles published in the literature developed in the LabTEVE and LEAPIG re-
search laboratories. Among the proposed classification methods is a with the Gamma distribu-
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Imports caTools, datasets, doParallel, e1071, EnvStats, foreach, MASS, 
mlbench, parallel, Rdpack, rootSolve, stats, trapezoid
RdMacros Rdpack
Depends R (>= 2.10)
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
URL https://github.com/Jodavid/FuzzyClass
BugReports https://github.com/Jodavid/FuzzyClass/issues
NeedsCompilation no
Author Jodavid Ferreira [aut, cre] (<https://orcid.org/0000-0002-2131-6464>),
Ronei Moraes [ctb] (<https://orcid.org/0000-0001-8436-8950>),
Arthur Ricardo [ctb]
Maintainer Jodavid Ferreira <jodavid@protonmail.com>
Repository CRAN
Date/Publication 2022-05-27 08:40:15 UTC
ExpNBFuzzyParam

R topics documented:

ExpNBFuzzyParam .................................................. 2
FuzzyBetaNaiveBayes ........................................ 3
FuzzyBinomialNaiveBayes .............................. 4
FuzzyExponentialNaiveBayes ......................... 5
FuzzyGammaNaiveBayes ......................................... 7
FuzzyGaussianNaiveBayes ....................................... 8
FuzzyNaiveBayes .................................................. 9
FuzzyPoissonNaiveBayes ..................................... 10
FuzzyTriangularNaiveBayes ............................. 11
GauNBFuzzyParam ............................................. 12
PoiNBFuzzyParam ............................................. 14
SimulatedData .................................................. 15
VirtualRealityData ............................................. 15

Index 16

ExpNBFuzzyParam  Fuzzy Exponential Naive Bayes Classifier with Fuzzy parameters

Description

ExpNBFuzzyParam Fuzzy Exponential Naive Bayes Classifier with Fuzzy parameters

Usage

ExpNBFuzzyParam(train, cl, metd = 1, cores = 2)

Arguments

train  matrix or data frame of training set cases.
cl  factor of true classifications of training set
metd  Method of transforming the triangle into scalar. It is the type of data entry for the test sample, use metd 1 if you want to use the Baricentro technique and use metd 2 if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis).
cores  how many cores of the computer do you want to use (default = 2)

Value

A vector of classifications

References

Examples

```r
set.seed(1) # determining a seed
data(VirtualRealityData)

# Splitting into Training and Testing
split <- caTools::sample.split(t(VirtualRealityData[, 1]), SplitRatio = 0.7)
Train <- subset(VirtualRealityData, split == "TRUE")
Test <- subset(VirtualRealityData, split == "FALSE")

# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_FENB <- ExpNBfuzzyParam(
  train = Train[, -4],
  cl = Train[, 4], metd = 1, cores = 2
)
pred_FENB <- predict(fit_FENB, test)
head(pred_FENB)
head(Test[, 4])
```

Description

**FuzzyBetaNaiveBayes** Fuzzy Beta Naive Bayes

Usage

```r
FuzzyBetaNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)
```

Arguments

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set
- **cores**: how many cores of the computer do you want to use (default = 2)
- **fuzzy**: boolean variable to use the membership function

Value

A vector of classifications
References


Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")

# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyBetaNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

head(pred_NBT)
head(Test[, 5])
```

FuzzyBinomialNaiveBayes

**FuzzyBinomialNaiveBayes**

**Fuzzy Binomial Naive Bayes**

**Description**

**FuzzyBinomialNaiveBayes** Fuzzy Binomial Naive Bayes

**Usage**

**FuzzyBinomialNaiveBayes**(train, cl, cores = 2, fuzzy = TRUE)

**Arguments**

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set
- **cores**: how many cores of the computer do you want to use (default = 2)
- **fuzzy**: boolean variable to use the membership function
FuzzyExponentialNaiveBayes

Value

A vector of classifications

References


Examples

set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rbinom(100, size = 10, prob = 0.2),
                     vari2 = rbinom(100, size = 10, prob = 0.2),
                     vari3 = rbinom(100, size = 10, prob = 0.2), class = 1)
class2 <- data.frame(vari1 = rbinom(100, size = 10, prob = 0.5),
                     vari2 = rbinom(100, size = 10, prob = 0.5),
                     vari3 = rbinom(100, size = 10, prob = 0.5), class = 2)
class3 <- data.frame(vari1 = rbinom(100, size = 10, prob = 0.8),
                     vari2 = rbinom(100, size = 10, prob = 0.8),
                     vari3 = rbinom(100, size = 10, prob = 0.8), class = 3)
data <- rbind(class1,class2,class3)

# Splitting into Training and Testing
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")

# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_NBT <- FuzzyBinomialNaiveBayes(
    train = Train[, -4],
    cl = Train[, 4], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

head(pred_NBT)
head(Test[, 4])

FuzzyExponentialNaiveBayes

Fuzzy Exponential Naive Bayes

Description

FuzzyExponentialNaiveBayes Fuzzy Exponential Naive Bayes
Usage

FuzzyExponentialNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

train         matrix or data frame of training set cases.
cl            factor of true classifications of training set
cores         how many cores of the computer do you want to use (default = 2)
fuzzy         boolean variable to use the membership function

Value

A vector of classifications

References


Examples

set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyExponentialNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

head(pred_NBT)
head(Test[, 5])
**FuzzyGammaNaiveBayes**

**Description**

FuzzyGammaNaiveBayes Fuzzy Gamma Naive Bayes

**Usage**

FuzzyGammaNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

**Arguments**

- `train`: matrix or data frame of training set cases.
- `cl`: factor of true classifications of training set
- `cores`: how many cores of the computer do you want to use (default = 2)
- `fuzzy`: boolean variable to use the membership function

**Value**

A vector of classifications

**References**


**Examples**

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")

# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyGammaNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)
```
FuzzyGaussianNaiveBayes

Gaussian Naive Bayes Classifier

Description

FuzzyGaussianNaiveBayes Gaussian Naive Bayes Classifier Zadeh-based

Usage

FuzzyGaussianNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

train  
matrix or data frame of training set cases.

cl  
factor of true classifications of training set

cores  
how many cores of the computer do you want to use (default = 2)

fuzzy  
boolean variable to use the membership function

Value

A vector of classifications

References


Examples

set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_GNB <- FuzzyGaussianNaiveBayes(

head(pred_NBT)
head(Test[, 5])
FuzzyNaiveBayes

```r
train = Train[, -5],
cl = Train[, 5], cores = 2
)
pred_GNB <- predict(fit_GNB, test)
head(pred_GNB)
head(Test[, 5])
```

---

FuzzyNaiveBayes  Fuzzy Naive Bayes

### Description

FuzzyNaiveBayes Fuzzy Naive Bayes

### Usage

FuzzyNaiveBayes(train, cl, fuzzy = TRUE, m = NULL, Pi = NULL)

### Arguments

- `train`: matrix or data frame of training set cases.
- `cl`: factor of true classifications of training set
- `fuzzy`: boolean variable to use the membership function
- `m`: is M/N, where M is the number of classes and N is the number of train lines
- `Pi`: is 1/M, where M is the number of classes

### Value

A vector of classifications

### References


### Examples

```r
set.seed(1) # determining a seed
data(HouseVotes84, package = "mlbench")

# Splitting into Training and Testing
split <- caTools::sample.split(t(HouseVotes84[, 1]), SplitRatio = 0.7)
Train <- subset(HouseVotes84, split == "TRUE")
Test <- subset(HouseVotes84, split == "FALSE")
```
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -1]
fit_FNB <- FuzzyNaiveBayes(
  train = Train[, -1],
  cl = Train[, 1]
)
pred_FNB <- predict(fit_FNB, test)

head(pred_FNB)
head(Test[, 1])

FuzzyPoissonNaiveBayes

Fuzzy Poisson Naive Bayes

Description

FuzzyPoissonNaiveBayes Fuzzy Poisson Naive Bayes

Usage

FuzzyPoissonNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

train matrix or data frame of training set cases.
cl factor of true classifications of training set
cores how many cores of the computer do you want to use (default = 2)
fuzzy boolean variable to use the membership function

Value

A vector of classifications

References

Examples

```r
set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rpois(100, lambda = 2),
                     vari2 = rpois(100, lambda = 2),
                     vari3 = rpois(100, lambda = 2), class = 1)
class2 <- data.frame(vari1 = rpois(100, lambda = 1),
                     vari2 = rpois(100, lambda = 1),
                     vari3 = rpois(100, lambda = 1), class = 2)
class3 <- data.frame(vari1 = rpois(100, lambda = 5),
                     vari2 = rpois(100, lambda = 5),
                     vari3 = rpois(100, lambda = 5), class = 3)
data <- rbind(class1, class2, class3)

# Splitting into Training and Testing
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_NBT <- FuzzyPoissonNaiveBayes(  
    train = Train[, -4],
    cl = Train[, 4], cores = 2
)
pred_NBT <- predict(fit_NBT, test)
head(pred_NBT)
head(Test[, 4])
```

FuzzyTriangularNaiveBayes

FuzzyTriangularNaiveBayes Naive Bayes Triangular Classifier

Description

FuzzyTriangularNaiveBayes Naive Bayes Triangular Classifier

Usage

FuzzyTriangularNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

- `train`: matrix or data frame of training set cases.
- `cl`: factor of true classifications of training set
- `cores`: how many cores of the computer do you want to use (default = 2)
- `fuzzy`: boolean variable to use the membership function
Value

A vector of classifications

References


Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# --------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyTriangularNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)
pred_NBT <- predict(fit_NBT, test)

head(pred_NBT)
head(Test[, 5])
```

---

**GauNBfuzzyParam**  
_Fuzzy Gaussian Naive Bayes Classifier with Fuzzy parameters_

**Description**

_GauNBfuzzyParam_ Fuzzy Gaussian Naive Bayes Classifier with Fuzzy parameters

**Usage**

_GauNBfuzzyParam_(train, cl, metd = 1, cores = 2)
**GauNB fuzzyParam**

**Arguments**

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set
- **metd**: Method of transforming the triangle into scalar. It is the type of data entry for the test sample, use metd 1 if you want to use the Baricentro technique and use metd 2 if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis).
- **cores**: how many cores of the computer do you want to use (default = 2)

**Value**

A vector of classifications

**References**


**Examples**

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[,1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_FGNB <- GauNBfuzzyParam(
  train = Train[, -5],
  cl = Train[, 5], metd = 1, cores = 2 )
pred_FGNB <- predict(fit_FGNB, test)

head(pred_FGNB)
head(Train[, 5])
```
**PoiNBuzzyParam**

**Fuzzy Poisson Naive Bayes Classifier with Fuzzy parameters**

**Description**

PoiNBuzzyParam Fuzzy Poisson Naive Bayes Classifier with Fuzzy parameters

**Usage**

PoiNBuzzyParam(train, cl, metd = 1, cores = 2)

**Arguments**

- `train`: matrix or data frame of training set cases.
- `cl`: factor of true classifications of training set.
- `metd`: Method of transforming the triangle into scalar, it is the type of data entry for the test sample, use `metd` 1 if you want to use the Baricentro technique and use `metd` 2 if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis).
- `cores`: how many cores of the computer do you want to use (default = 2)

**Value**

A vector of classifications

**References**


**Examples**

```r
set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rpois(100, lambda = 2),
                     vari2 = rpois(100, lambda = 2),
                     vari3 = rpois(100, lambda = 2), class = 1)
class2 <- data.frame(vari1 = rpois(100, lambda = 1),
                     vari2 = rpois(100, lambda = 1),
                     vari3 = rpois(100, lambda = 1), class = 2)
class3 <- data.frame(vari1 = rpois(100, lambda = 5),
                     vari2 = rpois(100, lambda = 5),
                     vari3 = rpois(100, lambda = 5), class = 3)
data <- rbind(class1, class2, class3)

# Splitting into Training and Testing
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
```
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")

# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
# test <- Test[, -4]
fit_FPoiNB <- PoiNBFuzzyParam(
  train = Train[, -4],
  cl = Train[, 4], metd = 1, cores = 2
)

pred_FPoiNB <- predict(fit_FPoiNB, test)

head(pred_FPoiNB)
head(Test[, 4])

---

**SimulatedData**  **Simulated Data**

**Description**
A dataset containing training data from Gammma Distribution

**Usage**
SimulatedData

**Format**
A dataset with 600 rows and 4 variables with 1 label.

---

**VirtualRealityData**  **Virtual Reality Simulator Data**

**Description**
A dataset containing training data from a virtual reality simulator

**Usage**
VirtualRealityData

**Format**
A dataset with 600 rows and 4 variables with 1 label.
Index

* datasets
  - SimulatedData, 15
  - VirtualRealityData, 15

ExpNBFuzzyParam, 2

FuzzyBetaNaiveBayes, 3
FuzzyBinomialNaiveBayes, 4
FuzzyExponentialNaiveBayes, 5
FuzzyGammaNaiveBayes, 7
FuzzyGaussianNaiveBayes, 8
FuzzyNaiveBayes, 9
FuzzyPoissonNaiveBayes, 10
FuzzyTriangularNaiveBayes, 11

GauNBFuzzyParam, 12

PoiNBFuzzyParam, 14

SimulatedData, 15

VirtualRealityData, 15