# Package 'FuzzyClass'

October 21, 2026

**Title** Fuzzy and Non-Fuzzy Classifiers

**Version** 0.3.0

**Description** It provides classifiers which can be used for discrete variables and for continuous variables based on the Naive Bayes and Fuzzy Naive Bayes hypothesis. Those methods were developed by researchers belong to the 'Laboratory of Technologies for Virtual Teaching and Statistics (LabTEVE)' and 'Laboratory of Applied Statistics to Image Processing and Geoprocessing (LEAPIG)' at 'Federal University of Paraiba, Brazil'. They considered some statistical distributions and their papers were published in the scientific literature, as for instance, the Gaussian classifier using fuzzy parameters, proposed by 'Moraes, Ferreira and Machado' (2021) <doi:10.1007/s40815-020-00936-4>. The classifiers are implemented as 'R6' classes and are also available as 'parsnip' models, so they can be used within the 'tidymodels' framework.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** caTools, cli (>= 3.4.0), e1071, EnvStats, MASS, mvtnorm, R6 (>= 2.5.0), Rdpack, rootSolve, stats, trapezoid

**RdMacros** Rdpack

**Depends** R (>= 4.1.0)

**Suggests** dials (>= 1.2.0), hardhat, knitr, maxLik, parsnip (>= 1.2.0), recipes, rlang (>= 1.1.0), rmarkdown, rsample, testthat (>= 3.0.0), tune (>= 1.2.0), workflows, yardstick

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**URL** https://leapigufpb.github.io/FuzzyClass/, https://github.com/leapigufpb/FuzzyClass

**BugReports** https://github.com/leapigufpb/FuzzyClass/issues

**Language** en-US

**Config/roxygen2/version** 8.1.0

**NeedsCompilation** no

**Author** Jodavid Ferreira [aut, cre] (ORCID:
<https://orcid.org/0000-0002-2131-6464>),
Ronei Moraes [ctb] (ORCID: <https://orcid.org/0000-0001-8436-8950>),
Liliane Machado [ctb] (ORCID: <https://orcid.org/0000-0002-1182-2929>),
Arthur Ricardo [ctb],
Isaac Araújo [ctb]

**Maintainer** Jodavid Ferreira <jodavid@protonmail.com>

**Repository** CRAN

**Date/Publication** 2026-10-21 20:50:02 UTC

# Contents

DWFuzzyGammaNaiveBayes . . . 3
DWFuzzyGammaNaiveBayesModel . . . 4
DWFuzzyHipergeometricNaiveBayes . . . 5
DWFuzzyHipergeometricNaiveBayesModel . . . 7
ExpNBFuzzyParam . . . 8
ExpNBFuzzyParamModel . . . 10
FuzzyBayesRule . . . 11
FuzzyBayesRuleModel . . . 12
FuzzyBetaNaiveBayes . . . 13
FuzzyBetaNaiveBayesModel . . . 15
FuzzyBinomialNaiveBayes . . . 16
FuzzyBinomialNaiveBayesModel . . . 17
FuzzyClassifier . . . 18
FuzzyExponentialNaiveBayes . . . 21
FuzzyExponentialNaiveBayesModel . . . 22
FuzzyGammaNaiveBayes . . . 23
FuzzyGammaNaiveBayesModel . . . 24
FuzzyGaussianNaiveBayes . . . 25
FuzzyGaussianNaiveBayesModel . . . 27
FuzzyGeoNaiveBayes . . . 28
FuzzyGeoNaiveBayesModel . . . 29
FuzzyHipergeometricNaiveBayes . . . 30
FuzzyHipergeometricNaiveBayesModel . . . 32
FuzzyNaiveBayes . . . 33
FuzzyNaiveBayesModel . . . 34
FuzzyPoissonNaiveBayes . . . 35
FuzzyPoissonNaiveBayesModel . . . 37
FuzzyTrapezoidalNaiveBayes . . . 38
FuzzyTrapezoidalNaiveBayesModel . . . 39
FuzzyTriangularNaiveBayes . . . 40
FuzzyTriangularNaiveBayesModel . . . 41
fuzzy_membership . . . 42
fuzzy_naive_bayes . . . 43
fuzzy_verbose . . . 45
GamWeightData . . . 46

GauNBFuzzyParam . . . . . 46
GauNBFuzzyParamModel . . . . . 48
HouseVotes84 . . . . . 49
PoiNBFuzzyParam . . . . . 50
PoiNBFuzzyParamModel . . . . . 52
predict.FuzzyClassifier . . . . . 53
SimulatedData . . . . . 54
VirtualRealityData . . . . . 54

**Index** **55**

---

`DWFuzzyGammaNaiveBayes`
*Double Weighted Fuzzy Gamma Naive Bayes*

---

## Description

`DWFuzzyGammaNaiveBayes` Double Weighted Fuzzy Gamma Naive Bayes

## Usage

```
DWFuzzyGammaNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  wdelta,
  weta,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `wdelta` | vector weight each class |
| `weta` | vector weight each feature |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained DWFuzzyGammaNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes RM, Soares EAMG, Machado LS (2020). "A double weighted fuzzy gamma naive bayes classifier." *Journal of Intelligent & Fuzzy Systems*, **38**(1), 577–588.

## See Also

DWFuzzyGammaNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"dw_gamma"`) for the tidymodels interface.

## Examples

```
set.seed(1) # determining a seed
data(GamWeightData)

# Splitting into Training and Testing
split <- caTools::sample.split(t(GamWeightData[, 1]), SplitRatio = 0.7)
Train <- subset(GamWeightData, split == "TRUE")
Test <- subset(GamWeightData, 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 <- DWFuzzyGammaNaiveBayes(
  train = Train[, -4],
  cl = Train[, 4], cores = 2,
  wdelta = c(2.002/6,1.998/6,2.000/6),
  weta = c(3/10,2/10, 5/10)
)

pred_NBT <- predict(fit_NBT, test)

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

---

`DWFuzzyGammaNaiveBayesModel` *Double Weighted Fuzzy Gamma Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Double Weighted Fuzzy Gamma Naive Bayes classifier. Create the model with `DWFuzzyGammaNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `DWFuzzyGammaNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `DWFuzzyGammaNaiveBayes`

## Methods

### Public methods:

- DWFuzzyGammaNaiveBayes$new()
- DWFuzzyGammaNaiveBayes$clone()

DWFuzzyGammaNaiveBayes$new(): Create a new (untrained) classifier.

*Usage:*

```
DWFuzzyGammaNaiveBayes$new(fuzzy = TRUE, wdelta, weta)
```

*Arguments:*

fuzzy See DWFuzzyGammaNaiveBayes().

wdelta See DWFuzzyGammaNaiveBayes().

weta See DWFuzzyGammaNaiveBayes().

DWFuzzyGammaNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DWFuzzyGammaNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- DWFuzzyGammaNaiveBayesModel$new(wdelta = rep(1/3, 3), weta = c(0.3, 0.2, 0.5))
data(GamWeightData)
model$fit(train = GamWeightData[, -4], cl = GamWeightData[, 4])
head(model$predict(GamWeightData[, -4]))
model
```

---

# DWFuzzyHipergeometricNaiveBayes

*Double Weighted Fuzzy Hipergeometric Naive Bayes*

---

## Description

DWFuzzyHipergeometricNaiveBayes Double Weighted Fuzzy Hipergeometric Naive Bayes

## Usage

```
DWFuzzyHipergeometricNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  wdelta,
  weta,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| fuzzy | boolean variable to use the membership function |
| wdelta | vector weight each class |
| weta | vector weight each feature |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained DWFuzzyHipergeometricNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Ferreira J, Machado LS, de Moraes RM (2025). "A New Double Weighted Fuzzy Hypergeometric Naive Bayes Network and its Application for User's Assessment in Virtual Reality Simulators." *International Journal of Computational Intelligence Systems*, **18**(1), 1–17.

## See Also

DWFuzzyHipergeometricNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"dw_hypergeometric"`) for the tidymodels interface.

## Examples

```
set.seed(1)

substituir_zero <- function(x) {
  if (x == 0) {
    while(1){
     new_valor <- rhyper(1,3,30,10)
      if(new_valor != 0){
        return(new_valor)
      }
    }
  } else {
    return(x)
  }
}

#Building dataframe
class1 <- data.frame(rhyper(72,3,30,10),
                     sample(seq(1,5),72,replace = TRUE), class = "Bem treinado")
colnames(class1)[1] <- "sucessos"
```

```
class1$sucessos <- sapply(class1$sucessos, substituir_zero)
colnames(class1)[2] <- "tentativas"
colnames(class1)[3] <- "avaliação"
class2 <- data.frame(rhyper(72,3,17,12),
                     sample(seq(4,7),72,replace = TRUE), class = "Mediano")
colnames(class2)[1] <- "sucessos"
#class2$sucessos <- ifelse(class2$sucessos == 0, 2, class2$sucessos)
colnames(class2)[2] <- "tentativas"
colnames(class2)[3] <- "avaliação"
class3 <- data.frame(rhyper(72,3,9,11),
                     sample(seq(7,10),72,replace = TRUE), class = "Precisa treinar")
colnames(class3)[1] <- "sucessos"
#class3$sucessos <- ifelse(class3$sucessos == 0, 3, class3$sucessos)
colnames(class3)[2] <- "tentativas"
colnames(class3)[3] <- "avaliação"
data <- rbind(class1,class2,class3)

# Weights
weta1 <- c(0.22, 0.33, 0.45)
wdelta1 <- c(0.33,0.33,0.33)

#spliting dataframe
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")

test <- Test[, -3]
fit_NHT <- DWFuzzyHipergeometricNaiveBayes(
 train = Train[, -3],
 cl = Train[, 3], cores = 2,
 fuzzy = TRUE,
 wdelta = wdelta1,
 weta = weta1)

pred_NHT <- predict(fit_NHT, test)
```

---

`DWFuzzyHipergeometricNaiveBayesModel`
*Double Weighted Fuzzy Hypergeometric Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Double Weighted Fuzzy Hypergeometric Naive Bayes classifier. Create the model with `DWFuzzyHipergeometricNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `DWFuzzyHipergeometricNaiveBayes()` trains and returns an object of this class.

## Super class

FuzzyClassifier -> DWFuzzyHipergeometricNaiveBayes

## Methods

### Public methods:

- DWFuzzyHipergeometricNaiveBayes$new()
- DWFuzzyHipergeometricNaiveBayes$clone()

DWFuzzyHipergeometricNaiveBayes$new(): Create a new (untrained) classifier.

*Usage:*

```
DWFuzzyHipergeometricNaiveBayes$new(fuzzy = TRUE, wdelta, weta)
```

*Arguments:*

fuzzy See DWFuzzyHipergeometricNaiveBayes().

wdelta See DWFuzzyHipergeometricNaiveBayes().

weta See DWFuzzyHipergeometricNaiveBayes().

DWFuzzyHipergeometricNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DWFuzzyHipergeometricNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- DWFuzzyHipergeometricNaiveBayesModel$new(wdelta = c(0.5, 0.5), weta = 1)
x <- data.frame(s1 = c(1, 2, 1, 2), n1 = c(5, 5, 6, 6))
y <- factor(c("A", "A", "B", "B"))
model$fit(train = x, cl = y)
model$predict(x)
model
```

---

# ExpNBFuzzyParam *Fuzzy Exponential Naive Bayes Classifier with Fuzzy parameters*

---

## Description

ExpNBFuzzyParam Fuzzy Exponential Naive Bayes Classifier with Fuzzy parameters

## Usage

```
ExpNBFuzzyParam(
  train,
  cl,
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6),
  cores = 2,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases |
| cl | factor of true classifications of training set |
| alphacut | value of the alpha-cut parameter, this value is between 0 and 1. |
| 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 Yager technique, 'metd=2' if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis), 'metd=3' if you want to use the Thorani technique, 'metd=4' if you want to use the alpha-order technique and 'metd=5' if you want to use the GTFN technique (article: A novel approach for arithmetic operations and ranking of generalized fuzzy numbers with application). |
| alp | When metd for 4, it is necessary to have alp which are alpha-cut defined |
| w | When metd for 4, it is necessary to have w which are alpha-cut weights defined |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained ExpNBFuzzyParamModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Rodrigues AK, Batista TV, Moraes RM, Machado LS (2016). "A new exponential naive bayes classifier with fuzzy parameters." In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1188–1194. IEEE.

## See Also

ExpNBFuzzyParamModel for the R6 interface and fuzzy_naive_bayes() (engine "exponential_fuzzy_param") for the tidymodels interface.

## Examples

```
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])
```

---

# ExpNBFuzzyParamModel *Exponential Naive Bayes with Fuzzy Parameters (R6 class)*

---

## Description

R6 implementation of the Exponential Naive Bayes with Fuzzy Parameters classifier. Create the model with `ExpNBFuzzyParamModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `ExpNBFuzzyParam()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `ExpNBFuzzyParam`

## Methods

### Public methods:

- `ExpNBFuzzyParam$new()`
- `ExpNBFuzzyParam$clone()`

`ExpNBFuzzyParam$new()`**:** Create a new (untrained) classifier.

*Usage:*

```
ExpNBFuzzyParam$new(
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6)
)
```

*Arguments:*

`alphacut` See `ExpNBFuzzyParam()`.

`metd` See `ExpNBFuzzyParam()`.

`alp` See `ExpNBFuzzyParam()`.

`w` See `ExpNBFuzzyParam()`.

`ExpNBFuzzyParam$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
ExpNBFuzzyParam$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- ExpNBFuzzyParamModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

`FuzzyBayesRule` *Fuzzy Bayes Rule*

---

## Description

`FuzzyBayesRule` Fuzzy Bayes Rule

## Usage

```
FuzzyBayesRule(train, cl, cores = 2, fuzzy = TRUE, verbose = fuzzy_verbose())
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyBayesRuleModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes R, Machado L (2008). "Fuzzy Bayes Rule for On-Line Training Assessment in Virtual Reality Simulators." *Multiple-Valued Logic and Soft Computing*, **14**, 325-338.

## See Also

FuzzyBayesRuleModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"bayes_rule"`) for the tidymodels interface.

## 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 <- FuzzyBayesRule(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

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

---

`FuzzyBayesRuleModel` *Fuzzy Bayes Rule (R6 class)*

---

## Description

R6 implementation of the Fuzzy Bayes Rule classifier. Create the model with `FuzzyBayesRuleModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyBayesRule()` trains and returns an object of this class.

### Super class

FuzzyClassifier -> FuzzyBayesRule

### Methods

#### Public methods:

- FuzzyBayesRule$new()
- FuzzyBayesRule$clone()

FuzzyBayesRule$new(): Create a new (untrained) classifier.

*Usage:*

```
FuzzyBayesRule$new(fuzzy = TRUE)
```

*Arguments:*

fuzzy See FuzzyBayesRule().

FuzzyBayesRule$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyBayesRule$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### Examples

```
model <- FuzzyBayesRuleModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

## FuzzyBetaNaiveBayes *Fuzzy Beta Naive Bayes*

---

### Description

FuzzyBetaNaiveBayes Fuzzy Beta Naive Bayes

### Usage

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

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| fuzzy | boolean variable to use the membership function |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained FuzzyBetaNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use predict(object, newdata, type = "class") or object$predict(newdata) to classify new observations; type = "prob" (or "matrix") returns the matrix of class probabilities.

## References

Moraes RM, Rodrigues AKG, Soares EAMG, Machado LS (2020). "A new fuzzy beta naive Bayes classifier." In *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*, 437–445. World Scientific.

## See Also

FuzzyBetaNaiveBayesModel for the R6 interface and fuzzy_naive_bayes() (engine "beta") for the tidymodels interface.

## 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 <- FuzzyBetaNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

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

FuzzyBetaNaiveBayesModel

*Fuzzy Beta Naive Bayes (R6 class)*

## Description

R6 implementation of the Fuzzy Beta Naive Bayes classifier. Create the model with `FuzzyBetaNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyBetaNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyBetaNaiveBayes`

## Methods

### Public methods:

- `FuzzyBetaNaiveBayes$new()`
- `FuzzyBetaNaiveBayes$clone()`

`FuzzyBetaNaiveBayes$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyBetaNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

`fuzzy` See `FuzzyBetaNaiveBayes()`.

`FuzzyBetaNaiveBayes$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyBetaNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- FuzzyBetaNaiveBayesModel$new()
model$fit(train = iris[, -5] / 10, cl = iris$Species)
head(model$predict(iris[, -5] / 10))
head(model$predict(iris[, -5] / 10, type = "prob"))
model
```

`FuzzyBinomialNaiveBayes` *Fuzzy Binomial Naive Bayes*

## Description

`FuzzyBinomialNaiveBayes` Fuzzy Binomial Naive Bayes

## Usage

```
FuzzyBinomialNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyBinomialNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes RM, Machado LS (2016). "A Fuzzy Binomial Naive Bayes classifier for epidemiological data." In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 745–750. IEEE.

## See Also

FuzzyBinomialNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"binomial"`) for the tidymodels interface.

## 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])
```

---

# FuzzyBinomialNaiveBayesModel

*Fuzzy Binomial Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Binomial Naive Bayes classifier. Create the model with `FuzzyBinomialNaiveBayesModel$n`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyBinomialNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyBinomialNaiveBayes`

## Methods

### Public methods:

- `FuzzyBinomialNaiveBayes$new()`

- FuzzyBinomialNaiveBayes$clone()

FuzzyBinomialNaiveBayes$new(): Create a new (untrained) classifier.

*Usage:*

```
FuzzyBinomialNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

fuzzy See FuzzyBinomialNaiveBayes().

FuzzyBinomialNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyBinomialNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- FuzzyBinomialNaiveBayesModel$new()
x <- round(iris[, -5])
model$fit(train = x, cl = iris$Species)
head(model$predict(x))
head(model$predict(x, type = "prob"))
model
```

---

# FuzzyClassifier *Base R6 class of the FuzzyClass classifiers*

---

## Description

FuzzyClassifier is the (abstract) R6 class shared by every classifier of the package. It implements the common workflow:

- $new(...) creates an untrained model holding its hyperparameters;
- $fit(train, cl) estimates the model and reports the progress;
- $predict(newdata, type) classifies new observations;
- $set_params(...) changes hyperparameters (the model must be re-trained).

Do not instantiate it directly: use one of the concrete classes, e.g. FuzzyGaussianNaiveBayesModel, FuzzyGammaNaiveBayesModel or GauNBFuzzyParamModel. Each of them also has a functional interface (e.g. FuzzyGaussianNaiveBayes()) that trains and returns the R6 object, and every classifier is available to tidymodels through fuzzy_naive_bayes().

Since $fit() and $set_params() return the object invisibly, calls can be chained: FuzzyGammaNaiveBayesModel$new()$fi
y)$predict(x_new).

## Public fields

`params` Named list with the hyperparameters of the classifier.

`estimates` List with the quantities estimated by `$fit()` (`NULL` while the model has not been trained).

`predictors` Names of the predictors seen during training.

`classes` Classes (levels of the response) seen during training.

`n_obs` Number of training observations.

`fit_time` Training time, in seconds.

## Active bindings

`label` Name of the classifier (read-only); it depends on whether the membership functions are used (`fuzzy`).

## Methods

### Public methods:

- `FuzzyClassifier$new()`
- `FuzzyClassifier$fit()`
- `FuzzyClassifier$predict()`
- `FuzzyClassifier$set_params()`
- `FuzzyClassifier$is_fitted()`
- `FuzzyClassifier$print()`
- `FuzzyClassifier$clone()`

`FuzzyClassifier$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyClassifier$new(...)
```

*Arguments:*

`...` Named hyperparameters of the classifier.

`FuzzyClassifier$fit()`: Estimate the classifier from training data.

*Usage:*

```
FuzzyClassifier$fit(train, cl, verbose = fuzzy_verbose())
```

*Arguments:*

`train` Matrix or data frame of training set cases.

`cl` Factor (or vector) of true classifications of the training set.

`verbose` Logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress.

*Returns:* The trained object, invisibly.

`FuzzyClassifier$predict()`: Classify new observations.

*Usage:*

```
FuzzyClassifier$predict(newdata, type = c("class", "prob", "matrix", "raw"))
```

*Arguments:*

`newdata` Matrix or data frame of test set cases, with the predictors in the same order used in training.

`type` `"class"` (default) for a factor of predicted classes, or `"prob"` for a matrix of class probabilities (one column per class). `"matrix"` and `"raw"` are accepted as synonyms of `"prob"`.

*Returns:* A factor (`type = "class"`) or a numeric matrix.

`FuzzyClassifier$set_params()`: Update hyperparameters. The model is reset and must be trained again with `$fit()`.

*Usage:*

```
FuzzyClassifier$set_params(...)
```

*Arguments:*

`...` Named hyperparameters, as in `$new()`.

*Returns:* The object, invisibly.

`FuzzyClassifier$is_fitted()`: Has the model been trained?

*Usage:*

```
FuzzyClassifier$is_fitted()
```

*Returns:* `TRUE` or `FALSE`.

`FuzzyClassifier$print()`: Print a summary of the classifier.

*Usage:*

```
FuzzyClassifier$print(...)
```

*Arguments:*

`...` Ignored.

`FuzzyClassifier$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyClassifier$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- FuzzyGaussianNaiveBayesModel$new(fuzzy = TRUE)
model$fit(train = iris[, -5], cl = iris$Species)
model

head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))

# S3 generics also work
head(predict(model, iris[, -5]))

# Change a hyperparameter and re-train, silently
model$set_params(fuzzy = FALSE)$fit(iris[, -5], iris$Species, verbose = FALSE)
model$label
```

`FuzzyExponentialNaiveBayes` *Fuzzy Exponential Naive Bayes*

## Description

`FuzzyExponentialNaiveBayes` Fuzzy Exponential Naive Bayes

## Usage

```
FuzzyExponentialNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyExponentialNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes RM, Machado LS (2016). "A fuzzy exponential naive bayes classifier." In *Uncertainty Modelling in Knowledge Engineering and Decision Making: Proceedings of the 12th International FLINS Conference*, 207–212. World Scientific.

## See Also

FuzzyExponentialNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"exponential"`) for the tidymodels interface.

## 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])
```

---

# FuzzyExponentialNaiveBayesModel

*Fuzzy Exponential Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Exponential Naive Bayes classifier. Create the model with `FuzzyExponentialNaiveBayesM` train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyExponentialNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyExponentialNaiveBayes`

## Methods

### Public methods:

- `FuzzyExponentialNaiveBayes$new()`
- `FuzzyExponentialNaiveBayes$clone()`

`FuzzyExponentialNaiveBayes$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyExponentialNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

`fuzzy` See `FuzzyExponentialNaiveBayes()`.

`FuzzyExponentialNaiveBayes$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyExponentialNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- FuzzyExponentialNaiveBayesModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

`FuzzyGammaNaiveBayes` *Fuzzy Gamma Naive Bayes*

---

## Description

`FuzzyGammaNaiveBayes` Fuzzy Gamma Naive Bayes

## Usage

```
FuzzyGammaNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyGammaNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes RM, Soares EAMG, Machado LS (2018). "A Fuzzy Gamma Naive Bayes Classifier." In *Data Science and Knowledge Engineering for Sensing Decision Support: Proceedings of the 13th International FLINS Conference (FLINS 2018)*, 691–699. World Scientific.

## See Also

FuzzyGammaNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"gamma"`) for the tidymodels interface.

## 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 <- FuzzyGammaNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

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

---

`FuzzyGammaNaiveBayesModel`

*Fuzzy Gamma Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Gamma Naive Bayes classifier. Create the model with `FuzzyGammaNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyGammaNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyGammaNaiveBayes`

## Methods

### Public methods:

- FuzzyGammaNaiveBayes$new()
- FuzzyGammaNaiveBayes$clone()

FuzzyGammaNaiveBayes$new(): Create a new (untrained) classifier.

*Usage:*

```
FuzzyGammaNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

fuzzy See FuzzyGammaNaiveBayes().

FuzzyGammaNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyGammaNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- FuzzyGammaNaiveBayesModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

# FuzzyGaussianNaiveBayes

*Fuzzy Gaussian Naive Bayes Classifier Zadeh-based*

---

## Description

FuzzyGaussianNaiveBayes Fuzzy Gaussian Naive Bayes Classifier Zadeh-based

## Usage

```
FuzzyGaussianNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| fuzzy | boolean variable to use the membership function |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained FuzzyGaussianNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use predict(object, newdata, type = "class") or object$predict(newdata) to classify new observations; type = "prob" (or "matrix") returns the matrix of class probabilities.

## References

Moraes RM, Machado LS (2012). "Online Assessment in Medical Simulators Based on Virtual Reality Using Fuzzy Gaussian Naive Bayes." *Journal of Multiple-Valued Logic & Soft Computing*, **18**.

## See Also

FuzzyGaussianNaiveBayesModel for the R6 interface and fuzzy_naive_bayes() (engine "gaussian") for the tidymodels interface.

## 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(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_GNB <- predict(fit_GNB, test)

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

FuzzyGaussianNaiveBayesModel

*Fuzzy Gaussian Naive Bayes (Zadeh-based) (R6 class)*

## Description

R6 implementation of the Fuzzy Gaussian Naive Bayes (Zadeh-based) classifier. Create the model with `FuzzyGaussianNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyGaussianNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyGaussianNaiveBayes`

## Methods

### Public methods:

- `FuzzyGaussianNaiveBayes$new()`
- `FuzzyGaussianNaiveBayes$clone()`

`FuzzyGaussianNaiveBayes$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyGaussianNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

`fuzzy` See `FuzzyGaussianNaiveBayes()`.

`FuzzyGaussianNaiveBayes$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyGaussianNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- FuzzyGaussianNaiveBayesModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

`FuzzyGeoNaiveBayes` *Fuzzy Naive Bayes Geometric Classifier*

## Description

`FuzzyGeoNaiveBayes` Naive Bayes Geometric Classifier

## Usage

```
FuzzyGeoNaiveBayes(train, cl, cores = 2, fuzzy = T, verbose = fuzzy_verbose())
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyGeoNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Ferreira J, Lopes A, Machado L, Moraes R (2023). "A Novel Fuzzy Geometric Naive Bayes Network for Online Skills Assessment in Training Based on Virtual Reality." *In Proceedings of the 15th International Joint Conference on Computational Intelligence*, 395–401.

## See Also

FuzzyGeoNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"geometric"`) for the tidymodels interface.

## Examples

```
set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rgeom(100,prob = 0.2),
                     vari2 = rgeom(100,prob = 0.2),
                     vari3 = rgeom(100,prob = 0.2), class = 1)
class2 <- data.frame(vari1 = rgeom(100,prob = 0.5),
                     vari2 = rgeom(100,prob = 0.5),
                     vari3 = rgeom(100,prob = 0.5), class = 2)
```

```
class3 <- data.frame(vari1 = rgeom(100,prob = 0.9),
                     vari2 = rgeom(100,prob = 0.9),
                     vari3 = rgeom(100,prob = 0.9), 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 <- FuzzyGeoNaiveBayes(
  train = Train[, -4],
  cl = Train[, 4], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

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

---

`FuzzyGeoNaiveBayesModel`

*Fuzzy Geometric Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Geometric Naive Bayes classifier. Create the model with `FuzzyGeoNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyGeoNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyGeoNaiveBayes`

## Methods

### Public methods:

- `FuzzyGeoNaiveBayes$new()`
- `FuzzyGeoNaiveBayes$clone()`

`FuzzyGeoNaiveBayes$new()`**:** Create a new (untrained) classifier.

*Usage:*

```
FuzzyGeoNaiveBayes$new(fuzzy = T)
```

*Arguments:*

fuzzy See FuzzyGeoNaiveBayes().

FuzzyGeoNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyGeoNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- FuzzyGeoNaiveBayesModel$new()
x <- round(iris[, -5])
model$fit(train = x, cl = iris$Species)
head(model$predict(x))
head(model$predict(x, type = "prob"))
model
```

---

FuzzyHipergeometricNaiveBayes

*Fuzzy Hipergeometric Naive Bayes*

---

## Description

FuzzyHipergeometricNaiveBayes Fuzzy Hipergeometric Naive Bayes

## Usage

```
FuzzyHipergeometricNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| fuzzy | boolean variable to use the membership function |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained FuzzyHipergeometricNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Ferreira J, Machado LS, de Moraes RM (2025). "A New Double Weighted Fuzzy Hypergeometric Naive Bayes Network and its Application for User's Assessment in Virtual Reality Simulators." *International Journal of Computational Intelligence Systems*, **18**(1), 1–17.

## See Also

FuzzyHipergeometricNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"hypergeometric"`) for the tidymodels interface.

## Examples

```
set.seed(1) # determining a seed

substituir_zero <- function(x) {
  if (x == 0) {
    while(1){
     new_valor <- rhyper(1,3,30,10)
      if(new_valor != 0){
        return(new_valor)
      }
    }
  } else {
    return(x)
  }
}

#Building dataframe
class1 <- data.frame(rhyper(72,3,30,10),
                     sample(seq(1,5),72,replace = TRUE), class = "Bem treinado")
colnames(class1)[1] <- "sucessos"
class1$sucessos <- sapply(class1$sucessos, substituir_zero)
colnames(class1)[2] <- "tentativas"
colnames(class1)[3] <- "avaliação"
class2 <- data.frame(rhyper(72,3,17,12),
                     sample(seq(4,7),72,replace = TRUE), class = "Mediano")
colnames(class2)[1] <- "sucessos"
colnames(class2)[2] <- "tentativas"
colnames(class2)[3] <- "avaliação"
class3 <- data.frame(rhyper(72,3,9,11),
                     sample(seq(7,10),72,replace = TRUE), class = "Precisa treinar")
colnames(class3)[1] <- "sucessos"
colnames(class3)[2] <- "tentativas"
colnames(class3)[3] <- "avaliação"
data <- rbind(class1,class2,class3)
```

```
#spliting dataframe
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")

test <- Test[, -3]
fit_NHT <- FuzzyHipergeometricNaiveBayes(
 train = Train[, -3],
 cl = Train[, 3], cores = 2
)

pred_NHT <- predict(fit_NHT, test)
```

---

FuzzyHipergeometricNaiveBayesModel

*Fuzzy Hypergeometric Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Hypergeometric Naive Bayes classifier. Create the model with `FuzzyHipergeometricNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyHipergeometricNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyHipergeometricNaiveBayes`

## Methods

### Public methods:

- `FuzzyHipergeometricNaiveBayes$new()`
- `FuzzyHipergeometricNaiveBayes$clone()`

`FuzzyHipergeometricNaiveBayes$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyHipergeometricNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

`fuzzy` See `FuzzyHipergeometricNaiveBayes()`.

`FuzzyHipergeometricNaiveBayes$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyHipergeometricNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- FuzzyHipergeometricNaiveBayesModel$new()
x <- data.frame(s1 = c(1, 2, 1, 2), n1 = c(5, 5, 6, 6))
y <- factor(c("A", "A", "B", "B"))
model$fit(train = x, cl = y)
model$predict(x)
model
```

---

# FuzzyNaiveBayes *Fuzzy Naive Bayes*

---

## Description

FuzzyNaiveBayes Fuzzy Naive Bayes

## Usage

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

## 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 |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained FuzzyNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use predict(object, newdata, type = "class") or object$predict(newdata) to classify new observations; type = "prob" (or "matrix") returns the matrix of class probabilities.

## References

Moraes RM, Machado LS (2009). "Another approach for fuzzy naive bayes applied on online training assessment in virtual reality simulators." In *Proceedings of Safety Health and Environmental World Congress*, 62–66.

## See Also

FuzzyNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"naive"`) for the tidymodels interface.

## Examples

```
set.seed(1) # determining a seed
data(HouseVotes84)

# 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])
```

---

`FuzzyNaiveBayesModel` *Fuzzy Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Naive Bayes classifier. Create the model with `FuzzyNaiveBayesModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyNaiveBayes`

## Methods

### Public methods:

- FuzzyNaiveBayes$new()
- FuzzyNaiveBayes$clone()

FuzzyNaiveBayes$new(): Create a new (untrained) classifier.

*Usage:*

```
FuzzyNaiveBayes$new(fuzzy = TRUE, m = NULL, Pi = NULL)
```

*Arguments:*

fuzzy See FuzzyNaiveBayes().

m See FuzzyNaiveBayes().

Pi See FuzzyNaiveBayes().

FuzzyNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- FuzzyNaiveBayesModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

# FuzzyPoissonNaiveBayes

*Fuzzy Poisson Naive Bayes*

---

## Description

FuzzyPoissonNaiveBayes Fuzzy Poisson Naive Bayes

## Usage

```
FuzzyPoissonNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| fuzzy | boolean variable to use the membership function |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained FuzzyPoissonNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use predict(object, newdata, type = "class") or object$predict(newdata) to classify new observations; type = "prob" (or "matrix") returns the matrix of class probabilities.

## References

Moraes RM, Machado LS (2015). "A fuzzy poisson naive bayes classifier for epidemiological purposes." In *2015 7th International Joint Conference on Computational Intelligence (IJCCI)*, volume 2, 193–198. IEEE.

## See Also

FuzzyPoissonNaiveBayesModel for the R6 interface and fuzzy_naive_bayes() (engine "poisson") for the tidymodels interface.

## Examples

```
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])
```

---

FuzzyPoissonNaiveBayesModel

*Fuzzy Poisson Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Poisson Naive Bayes classifier. Create the model with `FuzzyPoissonNaiveBayesModel$new`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyPoissonNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyPoissonNaiveBayes`

## Methods

### Public methods:

- `FuzzyPoissonNaiveBayes$new()`
- `FuzzyPoissonNaiveBayes$clone()`

`FuzzyPoissonNaiveBayes$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyPoissonNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

`fuzzy` See `FuzzyPoissonNaiveBayes()`.

`FuzzyPoissonNaiveBayes$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyPoissonNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- FuzzyPoissonNaiveBayesModel$new()
x <- round(iris[, -5])
model$fit(train = x, cl = iris$Species)
head(model$predict(x))
head(model$predict(x, type = "prob"))
model
```

`FuzzyTrapezoidalNaiveBayes`
*Fuzzy Naive Bayes Trapezoidal Classifier*

## Description

`FuzzyTrapezoidalNaiveBayes` Fuzzy Naive Bayes Trapezoidal Classifier

## Usage

```
FuzzyTrapezoidalNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = T,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyTrapezoidalNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Lopes A, Ferreira J, Machado LS, Moraes RM (2022). "A New Fuzzy Trapezoidal Naive Bayes Network as basis for Assessment in Training based on Virtual Reality." In *The 15th International FLINS Conference on Machine learning, Multi agent and Cyber physical systems (FLINS 2022)*. Nankai University.

## See Also

FuzzyTrapezoidalNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"trapezoidal"`) for the tidymodels interface.

## 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 <- FuzzyTrapezoidalNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

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

---

# FuzzyTrapezoidalNaiveBayesModel

*Fuzzy Trapezoidal Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Trapezoidal Naive Bayes classifier. Create the model with `FuzzyTrapezoidalNaiveBayesM` train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyTrapezoidalNaiveBayes()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `FuzzyTrapezoidalNaiveBayes`

## Methods

### Public methods:

- `FuzzyTrapezoidalNaiveBayes$new()`
- `FuzzyTrapezoidalNaiveBayes$clone()`

`FuzzyTrapezoidalNaiveBayes$new()`: Create a new (untrained) classifier.

*Usage:*

```
FuzzyTrapezoidalNaiveBayes$new(fuzzy = T)
```

*Arguments:*

`fuzzy` See `FuzzyTrapezoidalNaiveBayes()`.

`FuzzyTrapezoidalNaiveBayes$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyTrapezoidalNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- FuzzyTrapezoidalNaiveBayesModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

`FuzzyTriangularNaiveBayes`

*Fuzzy Naive Bayes Triangular Classifier*

---

## Description

`FuzzyTriangularNaiveBayes` Fuzzy Naive Bayes Triangular Classifier

## Usage

```
FuzzyTriangularNaiveBayes(
  train,
  cl,
  cores = 2,
  fuzzy = TRUE,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| `train` | matrix or data frame of training set cases. |
| `cl` | factor of true classifications of training set |
| `cores` | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| `fuzzy` | boolean variable to use the membership function |
| `verbose` | logical; if `TRUE` (default, see `fuzzy_verbose()`) a message reports the training progress. |

## Value

A trained FuzzyTriangularNaiveBayesModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes RM, Silva ILA, Machado LS (2020). "Online skills assessment in training based on virtual reality using a novel fuzzy triangular naive Bayes network." In *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*, 446–454. World Scientific.

## See Also

FuzzyTriangularNaiveBayesModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"triangular"`) for the tidymodels interface.

## 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 <- FuzzyTriangularNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)

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

---

`FuzzyTriangularNaiveBayesModel`

*Fuzzy Triangular Naive Bayes (R6 class)*

---

## Description

R6 implementation of the Fuzzy Triangular Naive Bayes classifier. Create the model with `FuzzyTriangularNaiveBayesMode` train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `FuzzyTriangularNaiveBayes()` trains and returns an object of this class.

## Super class

FuzzyClassifier -> FuzzyTriangularNaiveBayes

## Methods

### Public methods:

- FuzzyTriangularNaiveBayes$new()
- FuzzyTriangularNaiveBayes$clone()

FuzzyTriangularNaiveBayes$new(): Create a new (untrained) classifier.

*Usage:*

```
FuzzyTriangularNaiveBayes$new(fuzzy = TRUE)
```

*Arguments:*

fuzzy See FuzzyTriangularNaiveBayes().

FuzzyTriangularNaiveBayes$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
FuzzyTriangularNaiveBayes$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- FuzzyTriangularNaiveBayesModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

# fuzzy_membership *Tuning parameters of the fuzzy naive Bayes models*

---

## Description

Parameter objects, for the dials package, of the main arguments of fuzzy_naive_bayes(). They are used automatically by tune when an argument is marked with tune().

## Usage

```
fuzzy_membership(values = c(TRUE, FALSE))

alpha_cut(range = c(1e-04, 0.1), trans = NULL)

ranking_method(range = c(1L, 5L), trans = NULL)
```

### Arguments

| | |
|---|---|
| values | A logical vector of possible values. |
| range | A two-element vector with the bounds of the parameter. |
| trans | A trans object from the scales package, or NULL. |

### Details

- fuzzy_membership(): whether the membership functions are used (argument fuzzy).
- alpha_cut(): smallest alpha-cut of the fuzzy parameters (argument alphacut).
- ranking_method(): method used to rank the fuzzy numbers (argument metd): 1 = Yager, 2 = Q distance, 3 = Thorani, 4 = alpha-order, 5 = GTFN.

These functions require the dials package.

### Value

A param object of the dials package.

### Examples

```
fuzzy_membership()
alpha_cut()
ranking_method()
```

---

## fuzzy_naive_bayes *Fuzzy naive Bayes models for tidymodels*

---

### Description

fuzzy_naive_bayes() defines a classification model, estimated by one of the FuzzyClass classifiers, that can be used with the tidymodels ecosystem (parsnip, workflows, recipes, rsample, tune, yardstick, ...).

The classifier is chosen through the engine:

| Engine | Classifier | Main arguments |
|---|---|---|
| "gaussian" (default) | FuzzyGaussianNaiveBayes() | fuzzy |
| "gamma" | FuzzyGammaNaiveBayes() | fuzzy |
| "beta" | FuzzyBetaNaiveBayes() | fuzzy |
| "exponential" | FuzzyExponentialNaiveBayes() | fuzzy |
| "binomial" | FuzzyBinomialNaiveBayes() | fuzzy |
| "poisson" | FuzzyPoissonNaiveBayes() | fuzzy |
| "geometric" | FuzzyGeoNaiveBayes() | fuzzy |
| "trapezoidal" | FuzzyTrapezoidalNaiveBayes() | fuzzy |
| "triangular" | FuzzyTriangularNaiveBayes() | fuzzy |
| "hypergeometric" | FuzzyHipergeometricNaiveBayes() | fuzzy |

| | | |
|---|---|---|
| "bayes_rule" | FuzzyBayesRule() | fuzzy |
| "naive" | FuzzyNaiveBayes() | fuzzy |
| "dw_gamma" | DWFuzzyGammaNaiveBayes() | fuzzy |
| "dw_hypergeometric" | DWFuzzyHipergeometricNaiveBayes() | fuzzy |
| "gaussian_fuzzy_param" | GauNBFuzzyParam() | alpha_cut, ranking_method |
| "exponential_fuzzy_param" | ExpNBFuzzyParam() | alpha_cut, ranking_method |
| "poisson_fuzzy_param" | PoiNBFuzzyParam() | alpha_cut, ranking_method |

Main arguments that do not apply to the chosen engine are ignored. Any other argument of the classifier (e.g. wdelta and weta for the double weighted classifiers, alp and w for ranking_method = 4, or verbose) can be passed through parsnip::set_engine().

The predictors are passed to the classifier as they are (no dummy variables are created), so use a recipe to prepare them when needed. Each fit prints a training message; set options(FuzzyClass.verbose = FALSE) (see fuzzy_verbose()) or set_engine(..., verbose = FALSE) to silence it, for example when tuning.

This function requires the parsnip package.

## Usage

```
fuzzy_naive_bayes(
  mode = "classification",
  engine = "gaussian",
  fuzzy = NULL,
  alpha_cut = NULL,
  ranking_method = NULL
)

## S3 method for class 'fuzzy_naive_bayes'
update(
  object,
  parameters = NULL,
  fuzzy = NULL,
  alpha_cut = NULL,
  ranking_method = NULL,
  fresh = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| mode | A single character string for the type of model. The only possible value is "classification". |
| engine | A single character string specifying the classifier (see the table above). |
| fuzzy | Logical; whether the membership functions are used (argument fuzzy of the classifiers). Tunable with fuzzy_membership(). |

| | |
|---|---|
| alpha_cut | Smallest alpha-cut of the fuzzy parameters (argument alphacut). Tunable with alpha_cut(). |
| ranking_method | Integer from 1 to 5: method used to rank the fuzzy numbers (argument metd). Tunable with ranking_method(). |
| object | A fuzzy_naive_bayes model specification. |
| parameters | A 1-row data frame (or NULL) with the new values of the main arguments, as used by tune::finalize_model(). |
| fresh | Logical; replace all the arguments (TRUE) or only update the ones given (FALSE)? |
| ... | Engine-specific arguments to update. |

## Value

A model specification object with classes fuzzy_naive_bayes and model_spec.

## Examples

```
library(parsnip)

spec <- fuzzy_naive_bayes(fuzzy = TRUE) |>
  set_engine("gaussian")
spec

model_fit <- fit(spec, Species ~ ., data = iris)
predict(model_fit, iris[c(1, 51, 101), ])
predict(model_fit, iris[c(1, 51, 101), ], type = "prob")

# Fuzzy parameters with the GTFN ranking method
fuzzy_naive_bayes(ranking_method = 5) |>
  set_engine("gaussian_fuzzy_param") |>
  fit(Species ~ ., data = iris)

# Engine-specific arguments
data(GamWeightData)
fuzzy_naive_bayes() |>
  set_engine("dw_gamma", wdelta = rep(1/3, 3), weta = c(0.3, 0.2, 0.5)) |>
  fit(class ~ ., data = GamWeightData)
```

---

# fuzzy_verbose *Verbosity of the FuzzyClass classifiers*

---

## Description

By default, training a classifier prints a short progress message. Set options(FuzzyClass.verbose = FALSE) to silence it globally (useful, for example, when tuning many models with tidymodels), or use the argument verbose of $fit() and of the functional interfaces.

### Usage

```
fuzzy_verbose()
```

### Value

The value of the option `FuzzyClass.verbose` (default `TRUE`).

### Examples

```
fuzzy_verbose()
old <- options(FuzzyClass.verbose = FALSE)
fit <- FuzzyGaussianNaiveBayes(iris[, -5], iris$Species) # silent
options(old)
```

---

## GamWeightData *Gamma Weighted Data*

---

### Description

A dataset simulated containing training data from a Gamma Distribution

### Usage

```
GamWeightData
```

### Format

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

---

## GauNBFuzzyParam *Fuzzy Gaussian Naive Bayes Classifier with Fuzzy parameters*

---

### Description

`GauNBFuzzyParam` Fuzzy Gaussian Naive Bayes Classifier with Fuzzy parameters

### Usage

```
GauNBFuzzyParam(
  train,
  cl,
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6),
  cores = 2,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| alphacut | value of the alpha-cut parameter, this value is between 0 and 1. |
| 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 Yager technique, 'metd=2' if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis), 'metd=3' if you want to use the Thorani technique, 'metd=4' if you want to use the alpha-order technique and 'metd=5' if you want to use the GTFN technique (article: A novel approach for arithmetic operations and ranking of generalized fuzzy numbers with application). |
| alp | When metd for 4, it is necessary to have alp which are alpha-cut defined |
| w | When metd for 4, it is necessary to have w which are alpha-cut weights defined |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained GauNBFuzzyParamModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Moraes RM, Ferreira JA, Machado LS (2021). "A New Bayesian Network Based on Gaussian Naive Bayes with Fuzzy Parameters for Training Assessment in Virtual Simulators." *International Journal of Fuzzy Systems*, **23**(3), 849–861.

## See Also

GauNBFuzzyParamModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"gaussian_fuzzy_param"`) for the tidymodels interface.

## 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_FGNB <- GauNBFuzzyParam(
  train = Train[, -5],
  cl = Train[, 5], metd = 1, cores = 2
)

pred_FGNB <- predict(fit_FGNB, test)

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

---

GauNBFuzzyParamModel *Gaussian Naive Bayes with Fuzzy Parameters (R6 class)*

---

## Description

R6 implementation of the Gaussian Naive Bayes with Fuzzy Parameters classifier. Create the model with `GauNBFuzzyParamModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `GauNBFuzzyParam()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `GauNBFuzzyParam`

## Methods

### Public methods:

- `GauNBFuzzyParam$new()`
- `GauNBFuzzyParam$clone()`

`GauNBFuzzyParam$new()`**:** Create a new (untrained) classifier.

*Usage:*

```
GauNBFuzzyParam$new(
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6)
)
```

*Arguments:*

`alphacut` See `GauNBFuzzyParam()`.

`metd` See `GauNBFuzzyParam()`.

`alp` See `GauNBFuzzyParam()`.

`w` See `GauNBFuzzyParam()`.

`GauNBFuzzyParam$clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
GauNBFuzzyParam$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
model <- GauNBFuzzyParamModel$new()
model$fit(train = iris[, -5], cl = iris$Species)
head(model$predict(iris[, -5]))
head(model$predict(iris[, -5], type = "prob"))
model
```

---

`HouseVotes84` *United States Congressional Voting Records 1984*

---

## Description

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

## Usage

```
data(HouseVotes84)
```

## Format

A data frame with 435 observations on 17 variables:

1. Class Name: 2 (democrat, republican)
2. handicapped-infants: 2 (y,n)
3. water-project-cost-sharing: 2 (y,n)
4. adoption-of-the-budget-resolution: 2 (y,n)
5. physician-fee-freeze: 2 (y,n)
6. el-salvador-aid: 2 (y,n)
7. religious-groups-in-schools: 2 (y,n)
8. anti-satellite-test-ban: 2 (y,n)
9. aid-to-nicaraguan-contras: 2 (y,n)
10. mx-missile: 2 (y,n)
11. immigration: 2 (y,n)
12. synfuels-corporation-cutback: 2 (y,n)
13. education-spending: 2 (y,n)
14. superfund-right-to-sue: 2 (y,n)
15. crime: 2 (y,n)
16. duty-free-exports: 2 (y,n)

17 export-administration-act-south-africa: 2 (y,n)

## Source

- Source: Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc., ington, D.C., 1985
- Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)

These data have been taken from the UCI Repository Of Machine Learning Databases at

- https://archive.ics.uci.edu/datasets
- https://archive.ics.uci.edu/datasets

and were converted to R format by Friedrich Leisch.

## References

Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [https://archive.ics.uci.edu/datasets]. Irvine, CA: University of California, Department of Information and Computer Science.

## Examples

```
data(HouseVotes84)
summary(HouseVotes84)
```

---

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

---

## Description

`PoiNBFuzzyParam` Fuzzy Poisson Naive Bayes Classifier with Fuzzy parameters

## Usage

```
PoiNBFuzzyParam(
  train,
  cl,
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6),
  cores = 2,
  verbose = fuzzy_verbose()
)
```

## Arguments

| | |
|---|---|
| train | matrix or data frame of training set cases. |
| cl | factor of true classifications of training set |
| alphacut | value of the alpha-cut parameter, this value is between 0 and 1. |
| 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 Yager technique, 'metd=2' if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis), 'metd=3' if you want to use the Thorani technique, 'metd=4' if you want to use the alpha-order technique and 'metd=5' if you want to use the GTFN technique (article: A novel approach for arithmetic operations and ranking of generalized fuzzy numbers with application). |
| alp | When metd for 4, it is necessary to have alp which are alpha-cut defined |
| w | When metd for 4, it is necessary to have w which are alpha-cut weights defined |
| cores | kept for backward compatibility; prediction is vectorized and does not start parallel workers (default = 2) |
| verbose | logical; if TRUE (default, see fuzzy_verbose()) a message reports the training progress. |

## Value

A trained PoiNBFuzzyParamModel R6 object (which inherits from FuzzyClassifier). Use `predict(object, newdata, type = "class")` or `object$predict(newdata)` to classify new observations; `type = "prob"` (or `"matrix"`) returns the matrix of class probabilities.

## References

Soares E, Machado L, Moraes R (2016). "Assessment of poisson naive bayes classifier with fuzzy parameters using data from different statistical distributions." In *IV Bazilian Congress on Fuzzy Sistems (CBSF 2016)*, volume 1, 57–68.

## See Also

PoiNBFuzzyParamModel for the R6 interface and `fuzzy_naive_bayes()` (engine `"poisson_fuzzy_param"`) for the tidymodels interface.

## Examples

```
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])
```

---

`PoiNBFuzzyParamModel` *Poisson Naive Bayes with Fuzzy Parameters (R6 class)*

---

## Description

R6 implementation of the Poisson Naive Bayes with Fuzzy Parameters classifier. Create the model with `PoiNBFuzzyParamModel$new()`, train it with `$fit(train, cl)` and classify new observations with `$predict(newdata)`. The functional interface `PoiNBFuzzyParam()` trains and returns an object of this class.

## Super class

`FuzzyClassifier` -> `PoiNBFuzzyParam`

## Methods

### Public methods:

- `PoiNBFuzzyParam$new()`
- `PoiNBFuzzyParam$clone()`

`PoiNBFuzzyParam$new()`**:** Create a new (untrained) classifier.

*Usage:*

```
PoiNBFuzzyParam$new(
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6)
)
```

*Arguments:*

alphacut See PoiNBFuzzyParam().

metd See PoiNBFuzzyParam().

alp See PoiNBFuzzyParam().

w See PoiNBFuzzyParam().

`PoiNBFuzzyParam$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
PoiNBFuzzyParam$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
model <- PoiNBFuzzyParamModel$new()
x <- round(iris[, -5])
model$fit(train = x, cl = iris$Species)
head(model$predict(x))
head(model$predict(x, type = "prob"))
model
```

---

`predict.FuzzyClassifier`

*Predict method for FuzzyClass classifiers*

---

## Description

S3 interface to the `$predict()` method of the FuzzyClassifier R6 objects, so that `predict(model, newdata)` keeps working as in previous versions.

## Usage

```
## S3 method for class 'FuzzyClassifier'
predict(object, newdata, type = "class", ...)
```

## Arguments

| | |
|---|---|
| object | A trained FuzzyClassifier object. |
| newdata | Matrix or data frame of test set cases. |
| type | "class" (default) for predicted classes, or "prob" (synonyms "matrix" and "raw") for the matrix of class probabilities. |
| ... | Ignored. |

## Value

A factor (type = "class") or a numeric matrix.

## SimulatedData *Simulated Data*

### Description

A dataset containing training data from Gammma Distribuition

### 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**
GamWeightData, 46
HouseVotes84, 49
SimulatedData, 54
VirtualRealityData, 54

alpha_cut *(*fuzzy_membership*)*, 42
alpha_cut(), *45*

DWFuzzyGammaNaiveBayes, 3
DWFuzzyGammaNaiveBayes(), *4, 5*, *44*
DWFuzzyGammaNaiveBayesModel, *3, 4*, 4
DWFuzzyHipergeometricNaiveBayes, 5
DWFuzzyHipergeometricNaiveBayes(), *7, 8*, *44*
DWFuzzyHipergeometricNaiveBayesModel, *6*, 7

ExpNBFuzzyParam, 8
ExpNBFuzzyParam(), *10, 11*, *44*
ExpNBFuzzyParamModel, *9*, 10

fuzzy_membership, 42
fuzzy_membership(), *44*
fuzzy_naive_bayes, 43
fuzzy_naive_bayes(), *4*, *6*, *9*, *12*, *14*, *16*, *18*, *21*, *24*, *26*, *28*, *31*, *34*, *36*, *38*, *41, 42*, *47*, *51*
fuzzy_verbose, 45
fuzzy_verbose(), *3*, *6*, *9*, *11*, *14*, *16*, *19*, *21*, *23*, *26*, *28*, *30*, *33*, *36*, *38*, *40*, *44*, *47*, *51*
FuzzyBayesRule, 11
FuzzyBayesRule(), *12, 13*, *44*
FuzzyBayesRuleModel, *12*, 12
FuzzyBetaNaiveBayes, 13
FuzzyBetaNaiveBayes(), *15*, *43*
FuzzyBetaNaiveBayesModel, *14*, 15
FuzzyBinomialNaiveBayes, 16
FuzzyBinomialNaiveBayes(), *17, 18*, *43*
FuzzyBinomialNaiveBayesModel, *16*, 17
FuzzyClassifier, *3, 4*, *6*, *8–10*, *12–17*, 18, *21–24*, *26–29*, *31–34*, *36–39*, *41, 42*, *47, 48*, *51–53*
FuzzyExponentialNaiveBayes, 21
FuzzyExponentialNaiveBayes(), *22*, *43*
FuzzyExponentialNaiveBayesModel, *21*, 22
FuzzyGammaNaiveBayes, 23
FuzzyGammaNaiveBayes(), *24, 25*, *43*
FuzzyGammaNaiveBayesModel, *18*, *23, 24*, 24
FuzzyGaussianNaiveBayes, 25
FuzzyGaussianNaiveBayes(), *18*, *27*, *43*
FuzzyGaussianNaiveBayesModel, *18*, *26*, 27
FuzzyGeoNaiveBayes, 28
FuzzyGeoNaiveBayes(), *29, 30*, *43*
FuzzyGeoNaiveBayesModel, *28*, 29
FuzzyHipergeometricNaiveBayes, 30
FuzzyHipergeometricNaiveBayes(), *32*, *43*
FuzzyHipergeometricNaiveBayesModel, *31*, 32
FuzzyNaiveBayes, 33
FuzzyNaiveBayes(), *34, 35*, *44*
FuzzyNaiveBayesModel, *33, 34*, 34
FuzzyPoissonNaiveBayes, 35
FuzzyPoissonNaiveBayes(), *37*, *43*
FuzzyPoissonNaiveBayesModel, *36*, 37
FuzzyTrapezoidalNaiveBayes, 38
FuzzyTrapezoidalNaiveBayes(), *39*, *43*
FuzzyTrapezoidalNaiveBayesModel, *38*, 39
FuzzyTriangularNaiveBayes, 40
FuzzyTriangularNaiveBayes(), *41–43*
FuzzyTriangularNaiveBayesModel, *41*, 41

GamWeightData, 46
GauNBFuzzyParam, 46
GauNBFuzzyParam(), *44*, *48*
GauNBFuzzyParamModel, *18*, *47*, 48

HouseVotes84, 49

parsnip::set_engine(), *44*
PoiNBFuzzyParam, 50
PoiNBFuzzyParam(), *44*, *52, 53*
PoiNBFuzzyParamModel, *51*, 52
predict.FuzzyClassifier, 53

ranking_method *(*fuzzy_membership*)*, 42
ranking_method(), *45*

SimulatedData, 54

update.fuzzy_naive_bayes
*(*fuzzy_naive_bayes*)*, 43

VirtualRealityData, 54