Package 'naivebayes'

March 16, 2024

Type Package

Title High Performance Implementation of the Naive Bayes Algorithm

Version 1.0.0

Description In this implementation of the Naive Bayes classifier following class conditional distributions are available: 'Bernoulli', 'Categorical', 'Gaussian', 'Poisson', 'Multinomial' and non-parametric representation of the class conditional density estimated via Kernel Density Estimation. Implemented classifiers handle missing data and can take advantage of sparse data.

License GPL-2

URL https://github.com/majkamichal/naivebayes, https://majkamichal.github.io/naivebayes/

BugReports https://github.com/majkamichal/naivebayes/issues

Suggests knitr, Matrix

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Michal Majka [aut, cre] (<https://orcid.org/0000-0002-7524-8014>)

Maintainer Michal Majka <michalmajka@hotmail.com>

Repository CRAN

Date/Publication 2024-03-16 12:50:02 UTC

R topics documented:

`bernoulli_naive_bayes` *Bernoulli Naive Bayes Classifier*

Description

`bernoulli_naive_bayes` is used to fit the Bernoulli Naive Bayes model in which all class conditional distributions are assumed to be Bernoulli and be independent.

Usage

```
bernoulli_naive_bayes(x, y, prior = NULL, laplace = 0, ...)
```

Arguments

`x`	matrix with numeric 0-1 predictors (matrix or dgCMatrix from Matrix package).
`y`	class vector (character/factor/logical).
`prior`	vector with prior probabilities of the classes. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
`laplace`	value used for Laplace smoothing (additive smoothing). Defaults to 0 (no Laplace smoothing).
`...`	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on numeric 0-1 values and class conditional probabilities are modelled with the Bernoulli distribution.

The Bernoulli Naive Bayes is available in both, `naive_bayes` and `bernoulli_naive_bayes`. The latter provides more efficient performance though. Faster calculation times come from restricting the data to a numeric 0-1 matrix and taking advantage of linear algebra operations. Sparse matrices of class "dgCMatrix" (Matrix package) are supported in order to furthermore speed up calculation times.

The `bernoulli_naive_bayes` and `naive_bayes()` are equivalent when the latter uses "0"-"1" character matrix.

The missing values (NAs) are omited while constructing probability tables. Also, the corresponding predict function excludes all NAs from the calculation of posterior probabilities (an informative warning is always given).

Value

`bernoulli_naive_bayes` returns an object of class "bernoulli_naive_bayes" which is a list with following components:

`data`	list with two components: `x` (matrix with predictors) and `y` (class variable).
`levels`	character vector with values of the class variable.
`laplace`	amount of Laplace smoothing (additive smoothing).
`prob1`	matrix with class conditional probabilities for the value 1. Based on this matrix full probability tables can be constructed. Please, see `tables` and `coef`.
`prior`	numeric vector with prior probabilities.
`call`	the call that produced this object.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

`naive_bayes`, `predict.bernoulli_naive_bayes`, `plot.bernoulli_naive_bayes`, `tables`, `get_cond_dist`, `%class%`

Examples

```
# library(naivebayes)

### Simulate the data:
set.seed(1)
cols <- 10 ; rows <- 100 ; probs <- c("0" = 0.9, "1" = 0.1)
M <- matrix(sample(0:1, rows * cols,  TRUE, probs), nrow = rows, ncol = cols)
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE, prob = c(0.3,0.7)))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 0
```

```
### Train the Bernoulli Naive Bayes
bnb <- bernoulli_naive_bayes(x = M, y = y, laplace = laplace)
summary(bnb)

# Classification
head(predict(bnb, newdata = M, type = "class")) # head(bnb %class% M)

# Posterior probabilities
head(predict(bnb, newdata = M, type = "prob")) # head(bnb %prob% M)

# Parameter estimates
coef(bnb)


### Sparse data: train the Bernoulli Naive Bayes
library(Matrix)
M_sparse <- Matrix(M, sparse = TRUE)
class(M_sparse) # dgCMatrix

# Fit the model with sparse data
bnb_sparse <- bernoulli_naive_bayes(M_sparse, y, laplace = laplace)

# Classification
head(predict(bnb_sparse, newdata = M_sparse, type = "class"))

# Posterior probabilities
head(predict(bnb_sparse, newdata = M_sparse, type = "prob"))

# Parameter estimates
coef(bnb_sparse)


### Equivalent calculation with general naive_bayes function.
### (no sparse data support by naive_bayes)

# Make sure that the columns are factors with the 0-1 levels
df <- as.data.frame(lapply(as.data.frame(M), factor, levels = c(0,1)))
# sapply(df, class)

nb <- naive_bayes(df, y, laplace = laplace)
summary(nb)
head(predict(nb, type = "prob"))

# Obtain probability tables
tables(nb, which = "V1")
tables(bnb, which = "V1")

# Visualise class conditional Bernoulli distributions
plot(nb, "V1", prob = "conditional")
plot(bnb, which = "V1", prob = "conditional")

# Check the equivalence of the class conditional distributions
```

```
all(get_cond_dist(nb) == get_cond_dist(bnb))
```

---

coef *Extract Model Coefficients*

Description

coef is a generic function which extracts model coefficients from specialized Naive Bayes objects.

Usage

```
## S3 method for class 'bernoulli_naive_bayes'
coef(object, ...)

## S3 method for class 'multinomial_naive_bayes'
coef(object, ...)

## S3 method for class 'poisson_naive_bayes'
coef(object, ...)

## S3 method for class 'gaussian_naive_bayes'
coef(object, ...)
```

Arguments

object	object of class inheriting from "*_naive_bayes".
...	not used.

Value

Coefficients extracted from the specialised Naive Bayes objects in form of a data frame.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

bernoulli_naive_bayes, multinomial_naive_bayes, poisson_naive_bayes, gaussian_naive_bayes

Examples

```
data(iris)
y <- iris[[5]]
M <- as.matrix(iris[-5])

### Train the Gaussian Naive Bayes
gnb <- gaussian_naive_bayes(x = M, y = y)
```

```
### Extract coefficients
coef(gnb)

coef(gnb)[c(TRUE,FALSE)] # only means

coef(gnb)[c(FALSE,TRUE)] # only standard deviations
```

---

gaussian_naive_bayes *Gaussian Naive Bayes Classifier*

Description

gaussian_naive_bayes is used to fit the Gaussian Naive Bayes model in which all class conditional distributions are assumed to be Gaussian and be independent.

Usage

```
gaussian_naive_bayes(x, y, prior = NULL, ...)
```

Arguments

x	numeric matrix with metric predictors (matrix or dgCMatrix from Matrix package).
y	class vector (character/factor/logical).
prior	vector with prior probabilities of the classes. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on real values (numeric/integer) and class conditional probabilities are modelled with the Gaussian distribution.

The Gaussian Naive Bayes is available in both, naive_bayes and gaussian_naive_bayes.The latter provides more efficient performance though. Faster calculation times come from restricting the data to a matrix with numeric columns and taking advantage of linear algebra operations. Sparse matrices of class "dgCMatrix" (Matrix package) are supported in order to furthermore speed up calculation times.

The gaussian_naive_bayes and naive_bayes() are equivalent when the latter is used with usepoisson = FALSE and usekernel = FALSE; and a matrix/data.frame contains numeric columns.

The missing values (NAs) are omited during the estimation process. Also, the corresponding predict function excludes all NAs from the calculation of posterior probabilities (an informative warning is always given).

Value

gaussian_naive_bayes returns an object of class "gaussian_naive_bayes" which is a list with following components:

data	list with two components: x (matrix with predictors) and y (class variable).
levels	character vector with values of the class variable.
params	list with two matrices, first containing the class conditional means and the second containing the class conditional standard deviations.
prior	numeric vector with prior probabilities.
call	the call that produced this object.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, predict.gaussian_naive_bayes, plot.gaussian_naive_bayes, tables, get_cond_dist, %class%

Examples

```
# library(naivebayes)
set.seed(1)
cols <- 10 ; rows <- 100
M <- matrix(rnorm(rows * cols, 100, 15), nrow = rows, ncol = cols)
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE, prob = c(0.3,0.7)))
colnames(M) <- paste0("V", seq_len(ncol(M)))


### Train the Gaussian Naive Bayes
gnb <- gaussian_naive_bayes(x = M, y = y)
summary(gnb)

# Classification
head(predict(gnb, newdata = M, type = "class")) # head(gnb %class% M)

# Posterior probabilities
head(predict(gnb, newdata = M, type = "prob")) # head(gnb %prob% M)

# Parameter estimates
coef(gnb)


### Sparse data: train the Gaussian Naive Bayes
library(Matrix)
M_sparse <- Matrix(M, sparse = TRUE)
class(M_sparse) # dgCMatrix

# Fit the model with sparse data
```

```
gnb_sparse <- gaussian_naive_bayes(M_sparse, y)

# Classification
head(predict(gnb_sparse, newdata = M_sparse, type = "class"))

# Posterior probabilities
head(predict(gnb_sparse, newdata = M_sparse, type = "prob"))

# Parameter estimates
coef(gnb_sparse)


### Equivalent calculation with general naive_bayes function.
### (no sparse data support by naive_bayes)

nb <- naive_bayes(M, y)
summary(nb)
head(predict(nb, type = "prob"))

# Obtain probability tables
tables(nb, which = "V1")
tables(gnb, which = "V1")

# Visualise class conditional Gaussian distributions
plot(nb, "V1", prob = "conditional")
plot(gnb, which = "V1", prob = "conditional")

# Check the equivalence of the class conditional distributions
all(get_cond_dist(nb) == get_cond_dist(gnb))
```

`get_cond_dist` *Obtain names of class conditional distribution assigned to features*

Description

Auxiliary function for `"naive_bayes"`, `"*_naive_bayes"` and `"naive_bayes_tables"` objects for obtaining names of class conditional distributions assigned to the features.

Usage

```
get_cond_dist(object)
```

Arguments

`object`	object of class inheriting from `"naive_bayes"` or `"*_naive_bayes"` or `"naive_bayes_tables"`.

Value

vector with names of class conditional distributions assigned to the features.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, bernoulli_naive_bayes, multinomial_naive_bayes, poisson_naive_bayes, gaussian_naive_bayes, tables

Examples

```
data(iris)
nb <- naive_bayes(Species ~ ., data = iris)
get_cond_dist(nb) # <=> attr(nb$tables, "cond_dist")
get_cond_dist(tables(nb))
```

Infix operators *Predict Method for Family of Naive Bayes Objects*

Description

The infix operators %class% and %prob% are shorthands for performing classification and obtaining posterior probabilities, respectively.

Usage

```
lhs %class% rhs
lhs %prob% rhs
```

Arguments

lhs	object of class inheriting from "naive_bayes" and "*_naive_bayes" family.
rhs	dataframe or matrix for "naive_bayes" objects OR matrix for all "*_naive_bayes" objects.

Details

If lhs is of class inheriting from the family of the Naive Bayes objects and rhs is either dataframe or matrix then the infix operators %class% and %prob% are equivalent to:

- lhs %class% rhs <=> predict(lhs, newdata = rhs, type = "class", threshold = 0.001, eps = 0)
- lhs %prob% rhs <=> predict(lhs, newdata = rhs, type == "prob", threshold = 0.001, eps = 0)

Compared to predict(), both operators do not allow changing values of fine tuning parameters threshold and eps.

Value

- %class% returns factor with class labels corresponding to the maximal conditional posterior probabilities.
- %prob% returns a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

predict.naive_bayes, predict.bernoulli_naive_bayes, predict.multinomial_naive_bayes, predict.poisson_naive_bayes, predict.gaussian_naive_bayes, predict.nonparametric_naive_bayes

Examples

```
### Fit the model
nb <- naive_bayes(Species ~ ., iris)

newdata <- iris[1:5,-5] # Let's pretend

### Classification
nb %class% newdata
predict(nb, newdata, type = "class")

### Posterior probabilities
nb %prob% newdata
predict(nb, newdata, type = "prob")
```

---

multinomial_naive_bayes

Multinomial Naive Bayes Classifier

Description

multinomial_naive_bayes is used to fit the Multinomial Naive Bayes model.

Usage

```
multinomial_naive_bayes(x, y, prior = NULL, laplace = 0.5, ...)
```

Arguments

x	numeric matrix with integer predictors (matrix or dgCMatrix from Matrix package).
y	class vector (character/factor/logical).
prior	vector with prior probabilities of the classes. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
laplace	value used for Laplace smoothing (additive smoothing). Defaults to 0.5.
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, where the features represent frequencies generated by a multinomial distribution.

Sparse matrices of class "dgCMatrix" (Matrix package) are supported in order to speed up calculation times.

Please note that the Multinomial Naive Bayes is not available through the naive_bayes function.

Value

multinomial_naive_bayes returns an object of class "multinomial_naive_bayes" which is a list with following components:

data	list with two components: x (matrix with predictors) and y (class variable).
levels	character vector with values of the class variable.
laplace	amount of Laplace smoothing (additive smoothing).
params	matrix with class conditional parameter estimates.
prior	numeric vector with prior probabilities.
call	the call that produced this object.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

References

Manning, C.D., Raghavan, P., & Schütze, H. (2008). An Introduction to Information Retrieval. Cambridge: Cambridge University Press (Chapter 13). Available at https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

See Also

predict.multinomial_naive_bayes, tables, get_cond_dist, %class%, coef.multinomial_naive_bayes

Examples

```
# library(naivebayes)

### Simulate the data:
set.seed(1)
cols <- 3 # words
rows <- 10000 # all documents
rows_spam <- 100 # spam documents

prob_word_non_spam <- prop.table(runif(cols))
prob_word_spam <- prop.table(runif(cols))

M1 <- t(rmultinom(rows_spam, size = cols, prob = prob_word_spam))
M2 <- t(rmultinom(rows - rows_spam, size = cols, prob = prob_word_non_spam))
M <- rbind(M1, M2)
colnames(M) <- paste0("word", 1:cols) ; rownames(M) <- paste0("doc", 1:rows)
head(M)
y <- c(rep("spam", rows_spam), rep("non-spam", rows - rows_spam))

### Train the Multinomial Naive Bayes
laplace <- 1
mnb <- multinomial_naive_bayes(x = M, y = y, laplace = laplace)
summary(mnb)

# Classification
head(predict(mnb, newdata = M, type = "class")) # head(mnb %class% M)

# Posterior probabilities
head(predict(mnb, newdata = M, type = "prob")) # head(mnb %prob% M)

# Parameter estimates
coef(mnb)

# Compare
round(cbind(non_spam = prob_word_non_spam, spam = prob_word_spam), 3)



### Sparse data: train the Multinomial Naive Bayes
library(Matrix)
M_sparse <- Matrix(M, sparse = TRUE)
class(M_sparse) # dgCMatrix

# Fit the model with sparse data
mnb_sparse <- multinomial_naive_bayes(M_sparse, y, laplace = laplace)

# Classification
head(predict(mnb_sparse, newdata = M_sparse, type = "class"))

# Posterior probabilities
head(predict(mnb_sparse, newdata = M_sparse, type = "prob"))
```

```
# Parameter estimates
coef(mnb_sparse)
```

`naivebayes` *naivebayes*

Description

The **naivebayes** package presents an efficient implementation of the widely-used Naive Bayes classifier. It upholds three core principles: efficiency, user-friendliness, and reliance solely on Base R. By adhering to the latter principle, the package ensures stability and reliability without introducing external dependencies. This design choice maintains efficiency by leveraging the optimized routines inherent in Base R, many of which are programmed in high-performance languages like C/C++ or FORTRAN. By following these principles, the naivebayes package provides a reliable and efficient tool for Naive Bayes classification tasks, ensuring that users can perform their analyses effectively and with ease, even in the presence of missing data.

Details

The general `naive_bayes()` function is designed to determine the class of each feature in a dataset, and depending on user specifications, it can assume various distributions for each feature. It currently supports the following class conditional distributions:

- Categorical distribution for discrete features (with Bernoulli distribution as a special case for binary outcomes)
- Poisson distribution for non-negative integer features
- Gaussian distribution for continuous features
- non-parametrically estimated densities via Kernel Density Estimation for continuous features

In addition to the general Naive Bayes function, the package provides specialized functions for various types of Naive Bayes classifiers. The specialized functions are carefully optimized for efficiency, utilizing linear algebra operations to excel when handling dense matrices. Additionally, they can also exploit sparsity of matrices for enhanced performance:

- Bernoulli Naiive Bayes via `bernoulli_naive_bayes()`
- Multinomial Naive Bayes via `multinomial_naive_bayes()`
- Poisson Naive Bayes via `poisson_naive_bayes()`
- Gaussian Naive Bayes via `gaussian_naive_bayes()`
- Non-Parametric Naive Bayes via `nonparametric_naive_bayes()`

These specialized classifiers are tailored to different assumptions about the underlying data distributions, offering users versatile tools for classification tasks. Moreover, the package incorporates various helper functions aimed at enhancing the user experience. Notably, the model fitting functions provided by the package can effectively handle missing data, ensuring that users can utilize the classifiers even in the presence of incomplete information.

Extended documentation can be found on the website:

- https://majkamichal.github.io/naivebayes/

Bug reports:

- https://github.com/majkamichal/naivebayes/issues

Contact:

- <michalmajka@hotmail.com>

`naive_bayes` *Naive Bayes Classifier*

Description

`naive_bayes` is used to fit Naive Bayes model in which predictors are assumed to be independent within each class label.

Usage

```
## Default S3 method:
naive_bayes(x, y, prior = NULL, laplace = 0,
  usekernel = FALSE, usepoisson = FALSE, ...)

## S3 method for class 'formula'
naive_bayes(formula, data, prior = NULL, laplace = 0,
  usekernel = FALSE, usepoisson = FALSE,
  subset, na.action = stats::na.pass, ...)
```

Arguments

`x`	matrix or dataframe with categorical (character/factor/logical) or metric (numeric) predictors.
`y`	class vector (character/factor/logical).
`formula`	an object of class `"formula"` (or one that can be coerced to "formula") of the form: `class ~ predictors` (class has to be a factor/character/logical).
`data`	matrix or dataframe with categorical (character/factor/logical) or metric (numeric) predictors.
`prior`	vector with prior probabilities of the classes. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
`laplace`	value used for Laplace smoothing (additive smoothing). Defaults to 0 (no Laplace smoothing).
`usekernel`	logical; if `TRUE`, `density` is used to estimate the class conditional densities of metric predictors. This applies to vectors with class "numeric". For further details on interaction between `usekernel` and `usepoisson` parameters please see Note below.

usepoisson	logical; if TRUE, Poisson distribution is used to estimate the class conditional PMFs of integer predictors (vectors with class "integer").
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. By default (na.pass), missing values are not removed from the data and are then omited while constructing tables. Alternatively, na.omit can be used to exclude rows with at least one missing value before constructing tables.
...	other parameters to density when usekernel = TRUE (na.rm defaults to TRUE) (for instance adjust, kernel or bw).

Details

Numeric (metric) predictors are handled by assuming that they follow Gaussian distribution, given the class label. Alternatively, kernel density estimation can be used (usekernel = TRUE) to estimate their class-conditional distributions. Also, non-negative integer predictors (variables representing counts) can be modelled with Poisson distribution (usepoisson = TRUE); for further details please see Note below. Missing values are not included into constructing tables. Logical variables are treated as categorical (binary) variables.

Value

naive_bayes returns an object of class "naive_bayes" which is a list with following components:

data	list with two components: x (dataframe with predictors) and y (class variable).
levels	character vector with values of the class variable.
laplace	amount of Laplace smoothing (additive smoothing).
tables	list of tables. For each categorical predictor a table with class-conditional probabilities, for each integer predictor a table with Poisson mean (if usepoisson = TRUE) and for each metric predictor a table with a mean and standard deviation or density objects for each class. The object tables contains also an additional attribute "cond_dist" - a character vector with the names of conditional distributions assigned to each feature.
prior	numeric vector with prior probabilities.
usekernel	logical; TRUE, if the kernel density estimation was used for estimating class conditional densities of numeric variables.
usepoisson	logical; TRUE, if the Poisson distribution was used for estimating class conditional PMFs of non-negative integer variables.
call	the call that produced this object.

Note

The class "numeric" contains "double" (double precision floating point numbers) and "integer". Depending on the parameters usekernel and usepoisson different class conditional distributions are applied to columns in the dataset with the class "numeric":

- If `usekernel=FALSE` and `poisson=FALSE` then Gaussian distribution is applied to each "numeric" variable ("numeric"&"integer" or "numeric"&"double")
- If `usekernel=TRUE` and `poisson=FALSE` then kernel density estimation (KDE) is applied to each "numeric" variable ("numeric"&"integer" or "numeric"&"double")
- If `usekernel=FALSE` and `poisson=TRUE` then Gaussian distribution is applied to each "double" vector and Poisson to each "integer" vector. (Gaussian: "numeric" & "double"; Poisson: "numeric" & "integer")
- If `usekernel=TRUE` and `poisson=TRUE` then kernel density estimation (KDE) is applied to each "double" vector and Poisson to each "integer" vector. (KDE: "numeric" & "double"; Poisson: "numeric" & "integer")

By default `usekernel=FALSE` and `poisson=FALSE`, thus Gaussian is applied to each numeric variable.

On the other hand, "character", "factor" and "logical" variables are assigned to the Categorical distribution with Bernoulli being its special case.

Prior the model fitting the classes of columns in the data.frame "data" can be easily checked via:

- `sapply(data, class)`
- `sapply(data, is.numeric)`
- `sapply(data, is.double)`
- `sapply(data, is.integer)`

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

`predict.naive_bayes`, `plot.naive_bayes`, `tables`, `get_cond_dist`, `%class%`

Examples

```
### Simulate example data
n <- 100
set.seed(1)
data <- data.frame(class = sample(c("classA", "classB"), n, TRUE),
                   bern = sample(LETTERS[1:2], n, TRUE),
                   cat  = sample(letters[1:3], n, TRUE),
                   logical = sample(c(TRUE,FALSE), n, TRUE),
                   norm = rnorm(n),
                   count = rpois(n, lambda = c(5,15)))
train <- data[1:95, ]
test <- data[96:100, -1]


### 1) General usage via formula interface
nb <- naive_bayes(class ~ ., train)
summary(nb)
```

```
# Classification
predict(nb, test, type = "class")
nb %class% test

# Posterior probabilities
predict(nb, test, type = "prob")
nb %prob% test

# Helper functions
tables(nb, 1)
get_cond_dist(nb)

# Note: all "numeric" (integer, double) variables are modelled
#       with Gaussian distribution by default.


### 2) General usage via matrix/data.frame and class vector
X <- train[-1]
class <- train$class
nb2 <- naive_bayes(x = X, y = class)
nb2 %prob% test


### 3) Model continuous variables non-parametrically
###    via kernel density estimation (KDE)
nb_kde <- naive_bayes(class ~ ., train, usekernel = TRUE)
summary(nb_kde)
get_cond_dist(nb_kde)

nb_kde %prob% test

# Visualize class conditional densities
plot(nb_kde, "norm", arg.num = list(legend.cex = 0.9), prob = "conditional")
plot(nb_kde, "count", arg.num = list(legend.cex = 0.9), prob = "conditional")

### ?density and ?bw.nrd for further documentation

# 3.1) Change Gaussian kernel to biweight kernel
nb_kde_biweight <- naive_bayes(class ~ ., train, usekernel = TRUE,
                               kernel = "biweight")
nb_kde_biweight %prob% test
plot(nb_kde_biweight, c("norm", "count"),
     arg.num = list(legend.cex = 0.9), prob = "conditional")


# 3.2) Change "nrd0" (Silverman's rule of thumb) bandwidth selector
nb_kde_SJ <- naive_bayes(class ~ ., train, usekernel = TRUE,
                               bw = "SJ")
nb_kde_SJ %prob% test
plot(nb_kde_SJ, c("norm", "count"),
     arg.num = list(legend.cex = 0.9), prob = "conditional")
```

```
# 3.3) Adjust bandwidth
nb_kde_adjust <- naive_bayes(class ~ ., train, usekernel = TRUE,
                          adjust = 1.5)
nb_kde_adjust %prob% test
plot(nb_kde_adjust, c("norm", "count"),
     arg.num = list(legend.cex = 0.9), prob = "conditional")


### 4) Model non-negative integers with Poisson distribution
nb_pois <- naive_bayes(class ~ ., train, usekernel = TRUE, usepoisson = TRUE)
summary(nb_pois)
get_cond_dist(nb_pois)

# Posterior probabilities
nb_pois %prob% test

# Class conditional distributions
plot(nb_pois, "count", prob = "conditional")

# Marginal distributions
plot(nb_pois, "count", prob = "marginal")


## Not run:
vars <- 10
rows <- 1000000
y <- sample(c("a", "b"), rows, TRUE)

# Only categorical variables
X1 <- as.data.frame(matrix(sample(letters[5:9], vars * rows, TRUE),
                           ncol = vars))
nb_cat <- naive_bayes(x = X1, y = y)
nb_cat
system.time(pred2 <- predict(nb_cat, X1))

## End(Not run)
```

`nonparametric_naive_bayes` *Non-Parametric Naive Bayes Classifier*

Description

`nonparametric_naive_bayes` is used to fit the Non-Parametric Naive Bayes model in which all class conditional distributions are non-parametrically estimated using kernel density estimator and are assumed to be independent.

Usage

```
nonparametric_naive_bayes(x, y, prior = NULL, ...)
```

Arguments

x	matrix with metric predictors (only numeric matrix accepted).
y	class vector (character/factor/logical).
prior	vector with prior probabilities of the classes. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
...	other parameters to density (for instance adjust, kernel or bw).

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on real values (numeric/integer) and class conditional probabilities are estimated in a non-parametric way with the kernel density estimator (KDE). By default Gaussian kernel is used and the smoothing bandwidth is selected according to the Silverman's 'rule of thumb'. For more details, please see the references and the documentation of density and bw.nrd0.

The Non-Parametric Naive Bayes is available in both, naive_bayes() and nonparametric_naive_bayes(). The latter does not provide a substantial speed up over the general naive_bayes() function but it is meant to be more transparent and user friendly.

The nonparametric_naive_bayes and naive_bayes() are equivalent when the latter is used with usekernel = TRUE and usepoisson = FALSE; and a matrix/data.frame contains only numeric variables.

The missing values (NAs) are omitted during the estimation process. Also, the corresponding predict function excludes all NAs from the calculation of posterior probabilities (an informative warning is always given).

Value

nonparametric_naive_bayes returns an object of class "nonparametric_naive_bayes" which is a list with following components:

data	list with two components: x (matrix with predictors) and y (class variable).
levels	character vector with values of the class variable.
dens	nested list containing density objects for each feature and class.
prior	numeric vector with prior probabilities.
call	the call that produced this object.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

References

Silverman, B. W. (1986). Density Estimation for Statistics and Data Analysis. Chapman & Hall.

See Also

predict.nonparametric_naive_bayes, plot.nonparametric_naive_bayes, tables, get_cond_dist, %class%

Examples

```
# library(naivebayes)
data(iris)
y <- iris[[5]]
M <- as.matrix(iris[-5])

### Train the Non-Parametric Naive Bayes
nnb <- nonparametric_naive_bayes(x = M, y = y)
summary(nnb)
head(predict(nnb, newdata = M, type = "prob"))

###  Equivalent calculation with general naive_bayes function:
nb <- naive_bayes(M, y, usekernel = TRUE)
summary(nb)
head(predict(nb, type = "prob"))

### Change kernel
nnb_kernel <- nonparametric_naive_bayes(x = M, y = y, kernel = "biweight")
plot(nnb_kernel, 1, prob = "conditional")

### Adjust bandwidth
nnb_adjust <- nonparametric_naive_bayes(M, y, adjust = 1.5)
plot(nnb_adjust, 1, prob = "conditional")

### Change bandwidth selector
nnb_bw <- nonparametric_naive_bayes(M, y, bw = "SJ")
plot(nnb_bw, 1, prob = "conditional")

### Obtain tables with conditional densities
# tables(nnb, which = 1)
```

---

plot.bernoulli_naive_bayes

Plot Method for bernoulli_naive_bayes Objects

Description

Plot method for objects of class "bernoulli_naive_bayes" designed for a quick look at the class marginal distributions or class conditional distributions of 0-1 valued predictors.

Usage

```
## S3 method for class 'bernoulli_naive_bayes'
plot(x, which = NULL, ask = FALSE, arg.cat = list(),
     prob = c("marginal", "conditional"), ...)
```

Arguments

x	object of class inheriting from "bernoulli_naive_bayes".
which	variables to be plotted (all by default). This can be any valid indexing vector or vector containing names of variables.
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
arg.cat	other parameters to be passed as a named list to mosaicplot.
prob	character; if "marginal" then marginal distributions of predictor variables for each class are visualised and if "conditional" then the class conditional distributions of predictor variables are depicted. By default, prob="marginal".
...	not used.

Details

Class conditional or class conditional distributions are visualised by mosaicplot.

The parameter prob controls the kind of probabilities to be visualized for each individual predictor Xi. It can take on two values:

- "marginal": $P(Xi|class) * P(class)$
- "conditional": $P(Xi|class)$

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes,bernoulli_naive_bayes predict.bernoulli_naive_bayes, tables, get_cond_dist

Examples

```
# Simulate data
cols <- 10 ; rows <- 100 ; probs <- c("0" = 0.4, "1" = 0.1)
M <- matrix(sample(0:1, rows * cols,  TRUE, probs), nrow = rows, ncol = cols)
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE, prob = c(0.3,0.7)))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 0.5

# Train the Bernoulli Naive Bayes model
bnb <- bernoulli_naive_bayes(x = M, y = y, laplace = laplace)

# Visualize class marginal probabilities corresponding to the first feature
plot(bnb, which = 1)

# Visualize class conditional probabilities corresponding to the first feature
plot(bnb, which = 1, prob = "conditional")
```

plot.gaussian_naive_bayes *Plot Method for gaussian_naive_bayes Objects*

Description

Plot method for objects of class "gaussian_naive_bayes" designed for a quick look at the class marginal or conditional Gaussian distributions of metric predictors.

Usage

```
## S3 method for class 'gaussian_naive_bayes'
plot(x, which = NULL, ask = FALSE, legend = TRUE,
  legend.box = FALSE, arg.num = list(),
  prob = c("marginal", "conditional"), ...)
```

Arguments

x	object of class inheriting from "gaussian_naive_bayes".
which	variables to be plotted (all by default). This can be any valid indexing vector or vector containing names of variables.
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
legend	logical; if TRUE a legend will be be plotted.
legend.box	logical; if TRUE a box will be drawn around the legend.
arg.num	other parameters to be passed as a named list to matplot.
prob	character; if "marginal" then marginal distributions of predictor variables for each class are visualised and if "conditional" then the class conditional distributions of predictor variables are depicted. By default, prob="marginal".
...	not used.

Details

Class marginal and class conditional Gaussian distributions are visualised by matplot.

The parameter prob controls the kind of probabilities to be visualized for each individual predictor Xi. It can take on two values:

- "marginal": $P(Xi|class) * P(class)$
- "conditional": $P(Xi|class)$

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, gaussian_naive_bayes, predict.gaussian_naive_bayes, tables, get_cond_dist

Examples

```
data(iris)
y <- iris[[5]]
M <- as.matrix(iris[-5])

### Train the Gaussian Naive Bayes with custom prior
gnb <- gaussian_naive_bayes(x = M, y = y, prior = c(0.1,0.3,0.6))

# Visualize class marginal Gaussian distributions corresponding
# to the first feature
plot(gnb, which = 1)

# Visualize class conditional Gaussian distributions corresponding
# to the first feature
plot(gnb, which = 1, prob = "conditional")
```

---

plot.naive_bayes *Plot Method for naive_bayes Objects*

Description

Plot method for objects of class "naive_bayes" designed for a quick look at the class marginal distributions or class conditional distributions of predictor variables.

Usage

```
## S3 method for class 'naive_bayes'
plot(x, which = NULL, ask = FALSE, legend = TRUE,
  legend.box = FALSE, arg.num = list(), arg.cat = list(),
  prob = c("marginal", "conditional"), ...)
```

Arguments

x	object of class inheriting from "naive_bayes".
which	variables to be plotted (all by default). This can be any valid indexing vector or vector containing names of variables.
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
legend	logical; if TRUE a legend will be be plotted.
legend.box	logical; if TRUE a box will be drawn around the legend.
arg.num	other parameters to be passed as a named list to matplot.
arg.cat	other parameters to be passed as a named list to mosaicplot.
prob	character; if "marginal" then marginal distributions of predictor variables for each class are visualised and if "conditional" then the class conditional distributions of predictor variables are depicted. By default, prob="marginal".
...	not used.

Details

Probabilities are visualised by `matplot` (for numeric (metric) predictors) and `mosaicplot` (for categorical predictors). In case of non parametric estimation of densities, the bandwidths are reported for each class. Nothing is returned. For numeric (metric) predictors position of the legend can be adjusted changed via `arg.num(..., legend.position = "topright")`. `legend.position` can be one of `"topright"` `"topleft"`, `"bottomright"`, `"bottomleft"`. In order to adjust the legend size following argument can be used: `arg.num(..., legend.cex = 0.9)`.

The parameter `prob` controls the kind of probabilities to be visualized for each individual predictor Xi. It can take on two values:

- "marginal": $P(Xi|class) * P(class)$
- "conditional": $P(Xi|class)$

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

`naive_bayes`, `predict.naive_bayes`, `%class%`, `tables`, `get_cond_dist`

Examples

```
data(iris)
iris2 <- cbind(iris, New = sample(letters[1:3], 150, TRUE))

# Fit the model with custom prior probabilities
nb <- naive_bayes(Species ~ ., data = iris2, prior = c(0.1, 0.3, 0.6))

# Visualize marginal distributions of two predictors
plot(nb, which = c("Sepal.Width", "Sepal.Length"), ask = TRUE)

# Visualize class conditional distributions corresponding to the first predictor
# with customized settings
plot(nb, which = 1, ask = FALSE, prob = "conditional",
     arg.num = list(col = 1:3, lty = 1,
     main = "Naive Bayes Plot", legend.position = "topright",
     legend.cex = 0.55))

# Visualize class marginal distributions corresponding to the first predictor
# with customized settings
plot(nb, which = 1, ask = FALSE, prob = "marginal",
     arg.num = list(col = 1:3, lty = 1,
     main = "Naive Bayes Plot", legend.position = "topright",
     legend.cex = 0.55))

# Visualize class marginal distribution corresponding to the predictor "new"
# with custom colours
plot(nb, which = "New", arg.cat = list(color = gray.colors(3)))
```

plot.nonparametric_naive_bayes *Plot Method for nonparametric_naive_bayes Objects*

Description

Plot method for objects of class "nonparametric_naive_bayes" designed for a quick look at the estimated class marginal or class conditional densities of metric predictors.

Usage

```
## S3 method for class 'nonparametric_naive_bayes'
plot(x, which = NULL, ask = FALSE, legend = TRUE,
  legend.box = FALSE, arg.num = list(),
  prob = c("marginal", "conditional"), ...)
```

Arguments

x	object of class inheriting from "nonparametric_naive_bayes".
which	variables to be plotted (all by default). This can be any valid indexing vector or vector containing names of variables.
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
legend	logical; if TRUE a legend will be be plotted.
legend.box	logical; if TRUE a box will be drawn around the legend.
arg.num	other parameters to be passed as a named list to matplot.
prob	character; if "marginal" then marginal distributions of predictor variables for each class are visualised and if "conditional" then the class conditional distributions of predictor variables are depicted. By default, prob="marginal".
...	not used.

Details

Estimated class marginal or class conditional densities are visualised by matplot.

The parameter prob controls the kind of probabilities to be visualized for each individual predictor Xi. It can take on two values:

- "marginal": $P(Xi|class) * P(class)$
- "conditional": $P(Xi|class)$

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, nonparametric_naive_bayes predict.nonparametric_naive_bayes, tables, get_cond_dist

Examples

```
data(iris)
y <- iris[[5]]
M <- as.matrix(iris[-5])

### Train the Non-Parametric Naive Bayes with custom prior
prior <- c(0.1,0.3,0.6)
nnb <- nonparametric_naive_bayes(x = M, y = y, prior = prior)
nnb2 <- nonparametric_naive_bayes(x = M, y = y, prior = prior, adjust = 1.5)
nnb3 <- nonparametric_naive_bayes(x = M, y = y, prior = prior, bw = "ucv")

# Visualize estimated class conditional densities corresponding
# to the first feature
plot(nnb, which = 1, prob = "conditional")
plot(nnb2, which = 1, prob = "cond")
plot(nnb3, which = 1, prob = "c")

# Visualize estimated class marginal densities corresponding
# to the first feature
plot(nnb, which = 1)
plot(nnb2, which = 1)
plot(nnb3, which = 1)
```

---

plot.poisson_naive_bayes

Plot Method for poisson_naive_bayes Objects

Description

Plot method for objects of class "poisson_naive_bayes" designed for a quick look at the class marginal or class conditional Poisson distributions of non-negative integer predictors.

Usage

```
## S3 method for class 'poisson_naive_bayes'
plot(x, which = NULL, ask = FALSE, legend = TRUE,
  legend.box = FALSE, arg.num = list(),
  prob = c("marginal", "conditional"), ...)
```

Arguments

x	object of class inheriting from "poisson_naive_bayes".
which	variables to be plotted (all by default). This can be any valid indexing vector or vector containing names of variables.

ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
legend	logical; if TRUE a legend will be be plotted.
legend.box	logical; if TRUE a box will be drawn around the legend.
arg.num	other parameters to be passed as a named list to matplot.
prob	character; if "marginal" then marginal distributions of predictor variables for each class are visualised and if "conditional" then the class conditional distributions of predictor variables are depicted. By default, prob="marginal".
...	not used.

Details

Class marginal or class conditional Poisson distributions are visualised by matplot.

The parameter prob controls the kind of probabilities to be visualized for each individual predictor Xi. It can take on two values:

- "marginal": $P(Xi|class) * P(class)$
- "conditional": $P(Xi|class)$

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, poisson_naive_bayes, predict.poisson_naive_bayes, tables, get_cond_dist

Examples

```
cols <- 10 ; rows <- 100
M <- matrix(rpois(rows * cols, lambda = 3), nrow = rows, ncol = cols)
# is.integer(M) # [1] TRUE
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 0

### Train the Poisson Naive Bayes
pnb <- poisson_naive_bayes(x = M, y = y, laplace = laplace)

# Visualize class conditional Poisson distributions corresponding
# to the first feature
plot(pnb, which = 1, prob = "conditional")

# Visualize class marginal Poisson distributions corresponding
# to the first feature
plot(pnb, which = 1)
```

poisson_naive_bayes *Poisson Naive Bayes Classifier*

Description

poisson_naive_bayes is used to fit the Poisson Naive Bayes model in which all class conditional distributions are assumed to be Poisson and be independent.

Usage

```
poisson_naive_bayes(x, y, prior = NULL, laplace = 0, ...)
```

Arguments

x	numeric matrix with integer predictors (matrix or dgCMatrix from Matrix package).
y	class vector (character/factor/logical).
prior	vector with prior probabilities of the classes. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
laplace	value used for Laplace smoothing (additive smoothing). Defaults to 0 (no Laplace smoothing).
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on non-negative integers (numeric/integer) and class conditional probabilities are modelled with the Poisson distribution.

The Poisson Naive Bayes is available in both, naive_bayes and poisson_naive_bayes. The latter provides more efficient performance though. Faster calculation times come from restricting the data to an integer-valued matrix and taking advantage of linear algebra operations. Sparse matrices of class "dgCMatrix" (Matrix package) are supported in order to furthermore speed up calculation times.

The poisson_naive_bayes and naive_bayes() are equivalent when the latter is used with usepoisson = TRUE and usekernel = FALSE; and a matrix/data.frame contains only integer-valued columns.

The missing values (NAs) are omited during the estimation process. Also, the corresponding predict function excludes all NAs from the calculation of posterior probabilities (an informative warning is always given).

Value

poisson_naive_bayes returns an object of class "poisson_naive_bayes" which is a list with following components:

data	list with two components: x (matrix with predictors) and y (class variable).

levels	character vector with values of the class variable.
laplace	amount of Laplace smoothing (additive smoothing).
params	matrix containing class conditional means.
prior	numeric vector with prior probabilities.
call	the call that produced this object.

Note

When the parameter laplace is set to positive constant c then this amount is added to all counts. This leads to the ("global") Bayesian estimation with an improper prior. In each case, the estimate is the expected value of the posterior which is given by the gamma distribution with parameters: cell count + c and number of observations in the cell.

If in one cell there is a zero count and laplace = 0 then one pseudo-count is automatically to each such cell. This corresponds to the "local" Bayesian estimation with uniform prior.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

predict.poisson_naive_bayes, plot.poisson_naive_bayes, tables, get_cond_dist, %class%, coef.poisson_naive_bayes

Examples

```
library(naivebayes)

### Simulate the data:
set.seed(1)
cols <- 10 ; rows <- 100
M <- matrix(rpois(rows * cols, lambda = 3), nrow = rows, ncol = cols)
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE, prob = c(0.3,0.7)))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 0.5


### Train the Poisson Naive Bayes
pnb <- poisson_naive_bayes(x = M, y = y, laplace = laplace)
summary(pnb)

# Classification
head(predict(pnb, newdata = M, type = "class")) # head(pnb %class% M)

# Posterior probabilities
head(predict(pnb, newdata = M, type = "prob")) # head(pnb %prob% M)

# Parameter estimates
coef(pnb)
```

```
### Sparse data: train the Poisson Naive Bayes
library(Matrix)
M_sparse <- Matrix(M, sparse = TRUE)
class(M_sparse) # dgCMatrix

# Fit the model with sparse data
pnb_sparse <- poisson_naive_bayes(M_sparse, y, laplace = laplace)

# Classification
head(predict(pnb_sparse, newdata = M_sparse, type = "class"))

# Posterior probabilities
head(predict(pnb_sparse, newdata = M_sparse, type = "prob"))

# Parameter estimates
coef(pnb_sparse)


### Equivalent calculation with general naive_bayes function.
### (no sparse data support by naive_bayes)

nb <- naive_bayes(M, y, laplace = laplace, usepoisson = TRUE)
summary(nb)
head(predict(nb, type = "prob"))

# Obtain probability tables
tables(nb, which = "V1")
tables(pnb, which = "V1")

# Visualise class conditional Poisson distributions
plot(nb, "V1", prob = "conditional")
plot(pnb, which = "V1", prob = "conditional")

# Check the equivalence of the class conditional distributions
all(get_cond_dist(nb) == get_cond_dist(pnb))
```

`predict.bernoulli_naive_bayes` *Predict Method for bernoulli_naive_bayes Objects*

Description

Classification based on the Bernoulli Naive Bayes model.

Usage

```
## S3 method for class 'bernoulli_naive_bayes'
predict(object, newdata = NULL, type = c("class","prob"), ...)
```

Arguments

object	object of class inheriting from "bernoulli_naive_bayes".
newdata	matrix with numeric 0-1 predictors.
type	if "class", new data points are classified according to the highest posterior probabilities. If "prob", the posterior probabilities for each class are returned.
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on numeric 0-1 values and class conditional probabilities are modelled with the Bernoulli distribution.

Class posterior probabilities are calculated using the Bayes' rule under the assumption of independence of predictors. If no newdata is provided, the data from the object is used.

The Bernoulli Naive Bayes is available in both, naive_bayes and bernoulli_naive_bayes. The implementation of the specialized Naive Bayes provides more efficient performance though. The speedup comes from the restricting the data input to a numeric 0-1 matrix and performing the linear algebra as well as vectorized operations on it. In other words, the efficiency comes at cost of the flexibility.

The NAs in the newdata are not included into the calculation of posterior probabilities; and if present an informative warning is given.

The bernoulli_naive_bayes function is equivalent to the naive_bayes function with the numeric 0-1 matrix being coerced, for instance, to the "0"-"1" character matrix.

Value

predict.bernoulli_naive_bayes returns either a factor with class labels corresponding to the maximal conditional posterior probabilities or a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, bernoulli_naive_bayes, plot.bernoulli_naive_bayes, tables, get_cond_dist, %class%

Examples

```
cols <- 10 ; rows <- 100 ; probs <- c("0" = 0.4, "1" = 0.1)
M <- matrix(sample(0:1, rows * cols,  TRUE, probs), nrow = rows, ncol = cols)
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE, prob = c(0.3,0.7)))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 0.5

### Train the Bernoulli Naive Bayes
```

```
bnb <- bernoulli_naive_bayes(x = M, y = y, laplace = laplace)

### Classification
head(predict(bnb, newdata = M, type = "class"))
head(bnb %class% M)

### Posterior probabilities
head(predict(bnb, newdata = M, type = "prob"))
head(bnb %prob% M)
```

`predict.gaussian_naive_bayes` *Predict Method for gaussian_naive_bayes Objects*

Description

Classification based on the Gaussian Naive Bayes model.

Usage

```
## S3 method for class 'gaussian_naive_bayes'
predict(object, newdata = NULL, type = c("class","prob"),
  threshold = 0.001, eps = 0, ...)
```

Arguments

`object`	object of class inheriting from `"gaussian_naive_bayes"`.
`newdata`	matrix with metric predictors (only numeric matrix accepted).
`type`	if "class", new data points are classified according to the highest posterior probabilities. If "prob", the posterior probabilities for each class are returned.
`threshold`	value by which zero probabilities or probabilities within the epsilon-range corresponding to metric variables are replaced (zero probabilities corresponding to categorical variables can be handled with Laplace (additive) smoothing).
`eps`	value that specifies an epsilon-range to replace zero or close to zero probabilities by `threshold`. It applies to metric variables.
`...`	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on real values and class conditional probabilities are modelled with the Gaussian distribution.

Class posterior probabilities are calculated using the Bayes' rule under the assumption of independence of predictors. If no `newdata` is provided, the data from the object is used.

The Gaussian Naive Bayes is available in both, `naive_bayes` and `gaussian_naive_bayes`. The implementation of the specialized Naive Bayes provides more efficient performance though. The speedup comes from the restricting the data input to a numeric matrix and performing the linear

algebra as well vectorized operations on it. In other words, the efficiency comes at cost of the flexibility.

The NAs in the newdata are not included into the calculation of posterior probabilities; and if present an informative warning is given.

The `gaussian_naive_bayes` function is equivalent to the `naive_bayes` function with the numeric matrix or a data.frame containing only numeric variables.

Value

`predict.gaussian_naive_bayes` returns either a factor with class labels corresponding to the maximal conditional posterior probabilities or a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

`naive_bayes`, `gaussian_naive_bayes`, `plot.gaussian_naive_bayes`, `tables`, `get_cond_dist`, `%class%`

Examples

```
data(iris)
y <- iris[[5]]
M <- as.matrix(iris[-5])

### Train the Gaussian Naive Bayes
gnb <- gaussian_naive_bayes(x = M, y = y)

### Classification
head(predict(gnb, newdata = M, type = "class"))
head(gnb %class% M)

### Posterior probabilities
head(predict(gnb, newdata = M, type = "prob"))
head(gnb %prob% M)
```

---

predict.multinomial_naive_bayes

Predict Method for multinomial_naive_bayes Objects

Description

Classification based on the Multinomial Naive Bayes model.

Usage

```
## S3 method for class 'multinomial_naive_bayes'
predict(object, newdata = NULL, type = c("class","prob"), ...)
```

Arguments

object	object of class inheriting from "multinomial_naive_bayes".
newdata	matrix with non-negative integer predictors (only numeric matrix is accepted).
type	if "class", new data points are classified according to the highest posterior probabilities. If "prob", the posterior probabilities for each class are returned.
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, where the features represent the frequencies with which events have been generated by a multinomial distribution.

The Multinomial Naive Bayes is not available through the naive_bayes function.

The NAs in the newdata are not included into the calculation of posterior probabilities; and if present an informative warning is given.

Value

predict.multinomial_naive_bayes returns either a factor with class labels corresponding to the maximal conditional posterior probabilities or a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

References

McCallum, Andrew; Nigam, Kamal (1998). A comparison of event models for Naive Bayes text classification (PDF). AAAI-98 workshop on learning for text categorization. 752. http://www.cs.cmu.edu/~knigam/paper

See Also

multinomial_naive_bayes, tables, get_cond_dist, %class%, coef.multinomial_naive_bayes

Examples

```
### Simulate the data:
cols <- 10 ; rows <- 100
M <- matrix(sample(0:5, rows * cols,  TRUE), nrow = rows, ncol = cols)
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE, prob = c(0.3,0.7)))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 1

### Train the Multinomial Naive Bayes
```

```
mnb <- multinomial_naive_bayes(x = M, y = y, laplace = laplace)

# Classification
head(predict(mnb, newdata = M, type = "class"))
head(mnb %class% M)

# Posterior probabilities
head(predict(mnb, newdata = M, type = "prob"))
head(mnb %prob% M)
```

`predict.naive_bayes` *Predict Method for naive_bayes Objects*

Description

Classification based on Naive Bayes models.

Usage

```
## S3 method for class 'naive_bayes'
predict(object, newdata = NULL, type = c("class","prob"),
  threshold = 0.001, eps = 0, ...)
```

Arguments

`object`	object of class inheriting from `"naive_bayes"`.
`newdata`	matrix or dataframe with categorical (character/factor/logical) or metric (numeric) predictors.
`type`	if "class", new data points are classified according to the highest posterior probabilities. If "prob", the posterior probabilities for each class are returned.
`threshold`	value by which zero probabilities or probabilities within the epsilon-range corresponding to metric variables are replaced (zero probabilities corresponding to categorical variables can be handled with Laplace (additive) smoothing).
`eps`	value that specifies an epsilon-range to replace zero or close to zero probabilities by `threshold`. It applies to metric variables.
`...`	not used.

Details

Computes conditional posterior probabilities for each class label using the Bayes' rule under the assumption of independence of predictors. If no new data is provided, the data from the object is used. Logical variables are treated as categorical (binary) variables. Predictors with missing values are not included into the computation of posterior probabilities.

Value

`predict.naive_bayes` returns either a factor with class labels corresponding to the maximal conditional posterior probabilities or a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, plot.naive_bayes, tables, get_cond_dist, %class%

Examples

```
### Simulate example data
n <- 100
set.seed(1)
data <- data.frame(class = sample(c("classA", "classB"), n, TRUE),
                   bern = sample(LETTERS[1:2], n, TRUE),
                   cat  = sample(letters[1:3], n, TRUE),
                   logical = sample(c(TRUE,FALSE), n, TRUE),
                   norm = rnorm(n),
                   count = rpois(n, lambda = c(5,15)))
train <- data[1:95, ]
test <- data[96:100, -1]

### Fit the model with default settings
nb <- naive_bayes(class ~ ., train)

# Classification
predict(nb, test, type = "class")
nb %class% test

# Posterior probabilities
predict(nb, test, type = "prob")
nb %prob% test


## Not run:
vars <- 10
rows <- 1000000
y <- sample(c("a", "b"), rows, TRUE)

# Only categorical variables
X1 <- as.data.frame(matrix(sample(letters[5:9], vars * rows, TRUE),
                           ncol = vars))
nb_cat <- naive_bayes(x = X1, y = y)
nb_cat
system.time(pred2 <- predict(nb_cat, X1))

## End(Not run)
```

predict.nonparametric_naive_bayes *Predict Method for nonparametric_naive_bayes Objects*

Description

Classification based on the Non-Parametric Naive Bayes model.

Usage

```
## S3 method for class 'nonparametric_naive_bayes'
predict(object, newdata = NULL, type = c("class","prob"),
  threshold = 0.001, eps = 0, ...)
```

Arguments

object	object of class inheriting from "nonparametric_naive_bayes".
newdata	matrix with metric predictors (only numeric matrix accepted).
type	if "class", new data points are classified according to the highest posterior probabilities. If "prob", the posterior probabilities for each class are returned.
threshold	value by which zero probabilities or probabilities within the epsilon-range corresponding to metric variables are replaced (zero probabilities corresponding to categorical variables can be handled with Laplace (additive) smoothing).
eps	value that specifies an epsilon-range to replace zero or close to zero probabilities by threshold. It applies to metric variables.
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features take on real values (numeric/integer) and class conditional probabilities are non-parametrically estimated with kernel density estimator. By default Gaussian kernel is used and the smoothing bandwidth is selected according to the Silverman's 'rule of thumb'. For more details, please see the references and the documentation of density and bw.nrd0.

The Non-Parametric Naive Bayes is available in both, naive_bayes() and nonparametric_naive_bayes(). This specialized implementation of the Naive Bayes does not provide a substantial speed-up over the general naive_bayes() function but it should be more transparent and user friendly.

The nonparametric_naive_bayes function is equivalent to naive_bayes() when the numeric matrix or a data.frame contains only numeric variables and usekernel = TRUE.

The missing values (NAs) are omitted during the parameter estimation. The NAs in the newdata in predict.nonparametric_naive_bayes() are not included into the calculation of posterior probabilities; and if present an informative warning is given.

Value

predict.nonparametric_naive_bayes returns either a factor with class labels corresponding to the maximal conditional posterior probabilities or a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

References

Silverman, B. W. (1986). Density Estimation for Statistics and Data Analysis. Chapman & Hall.

See Also

naive_bayes, nonparametric_naive_bayes, plot.nonparametric_naive_bayes, tables, get_cond_dist, naive_bayes, %class%

Examples

```
data(iris)
y <- iris[[5]]
M <- as.matrix(iris[-5])

### Train the Non-Parametric Naive Bayes
nnb <- nonparametric_naive_bayes(x = M, y = y, bw = "SJ")

### Classification
head(predict(nnb, newdata = M, type = "class"))
head(nnb %class% M)

### Posterior probabilities
head(predict(nnb, newdata = M, type = "prob"))
head(nnb %prob% M)
```

---

predict.poisson_naive_bayes

Predict Method for poisson_naive_bayes Objects

Description

Classification based on the Poisson Naive Bayes model.

Usage

```
## S3 method for class 'poisson_naive_bayes'
predict(object, newdata = NULL, type = c("class","prob"),
  threshold = 0.001, eps = 0, ...)
```

Arguments

object	object of class inheriting from "poisson_naive_bayes".
newdata	matrix with non-negative integer predictors (only numeric matrix is accepted).
type	if "class", new data points are classified according to the highest posterior probabilities. If "prob", the posterior probabilities for each class are returned.
threshold	value by which zero probabilities or probabilities within the epsilon-range corresponding to metric variables are replaced (zero probabilities corresponding to categorical variables can be handled with Laplace (additive) smoothing).
eps	value that specifies an epsilon-range to replace zero or close to zero probabilities by threshold.
...	not used.

Details

This is a specialized version of the Naive Bayes classifier, in which all features are non-negative integers and class conditional probabilities are modelled with the Poisson distribution.

Class posterior probabilities are calculated using the Bayes' rule under the assumption of independence of predictors. If no newdata is provided, the data from the object is used.

The Poisson Naive Bayes is available in both, naive_bayes and poisson_naive_bayes. The implementation of the specialized Naive Bayes provides more efficient performance though. The speedup comes from the restricting the data input to a numeric matrix and performing the linear algebra as well vectorized operations on it.

The NAs in the newdata are not included into the calculation of posterior probabilities; and if present an informative warning is given.

The poisson_naive_bayes function is equivalent to the naive_bayes function with usepoisson=TRUE and a numeric matrix or a data.frame containing only non-negative integer valued features (each variable has class "integer").

Value

predict.poisson_naive_bayes returns either a factor with class labels corresponding to the maximal conditional posterior probabilities or a matrix with class label specific conditional posterior probabilities.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

poisson_naive_bayes, plot.poisson_naive_bayes, tables, get_cond_dist, %class%, coef.poisson_naive_bayes

Examples

```
cols <- 10 ; rows <- 100
M <- matrix(rpois(rows * cols, lambda = 3), nrow = rows, ncol = cols)
# is.integer(M) # [1] TRUE
y <- factor(sample(paste0("class", LETTERS[1:2]), rows, TRUE))
colnames(M) <- paste0("V", seq_len(ncol(M)))
laplace <- 0

### Train the Poisson Naive Bayes
pnb <- poisson_naive_bayes(x = M, y = y, laplace = laplace)

### Classification
head(predict(pnb, newdata = M, type = "class"))
head(pnb %class% M)

### Posterior probabilities
head(predict(pnb, newdata = M, type = "prob"))
head(pnb %prob% M)
```

---

tables *Browse Tables of Naive Bayes Classifier*

Description

Auxiliary function for "naive_bayes" and "*_naive_bayes" objects for easy browsing tables.

Usage

```
tables(object, which = NULL)
```

Arguments

object	object of class inheriting from: "naive_bayes" and "*_naive_bayes".
which	tables to be showed (all by default). This can be any valid indexing vector or vector containing names of variables.

Details

Default print method for "naive_bayes" and "*_naive_bayes" objects shows at most five first tables. This auxiliary function tables returns by default all tables and allows easy subsetting via indexing variables.

Value

list with tables.

Author(s)

Michal Majka, <michalmajka@hotmail.com>

See Also

naive_bayes, bernoulli_naive_bayes, multinomial_naive_bayes, poisson_naive_bayes, gaussian_naive_bayes, tables, get_cond_dist

Examples

```
data(iris)
nb <- naive_bayes(Species ~ ., data = iris)
tables(nb, "Sepal.Length")
tables(nb, c("Sepal.Length", "Sepal.Width"))
tabs <- tables(nb, 1:2)
tabs
tabs[1]
```

Index