Package: bnclassify (via r-universe)

October 12, 2026

Title Learning Discrete Bayesian Network Classifiers from Data

Description State-of-the art algorithms for learning discrete Bayesian network classifiers from data, including a number of those described in Bielza & Larranaga (2014) <doi:10.1145/2576868>, with functions for prediction, model evaluation and inspection.

Version 0.4.8

URL https://github.com/bmihaljevic/bnclassify

BugReports https://github.com/bmihaljevic/bnclassify/issues

Depends R (>= 3.2.0)

Imports assertthat (>= 0.1), entropy(>= 1.2.0), matrixStats(>= 0.14.0), rpart(>= 4.1-8), Rcpp,

Suggests igraph, gRain(>= 1.2-3), gRbase(>= 1.7-0.1), mlr(>= 2.2), testthat(>= 0.8.1), knitr(>= 1.10.5), ParamHelpers(>= 1.5), rmarkdown(>= 0.7), mlbench, covr

Encoding UTF-8

License GPL (>= 2)

Maintainer Mihaljevic Bojan <boki.mihaljevic@gmail.com>

VignetteBuilder knitr

LinkingTo Rcpp, BH

RoxygenNote 7.3.1

Repository https://bmihaljevic.r-universe.dev

Date/Publication 2024-03-15 10:22:30 UTC

RemoteUrl https://github.com/bmihaljevic/bnclassify

RemoteRef HEAD

RemoteSha d121dfa3353a41c532c8fd84380b5e1eaa5a61d5

Contents

accuracy . . . 2
aode . . . 3
as_mlr . . . 3
bnc . . . 4
bnc_bn . . . 5
bnc_dag . . . 5
bnclassify . . . 6
car . . . 8
cmi . . . 8
cv . . . 9
grain_and_graph . . . 10
greedy_wrapper . . . 10
inspect_bnc_bn . . . 12
inspect_bnc_dag . . . 13
learn_params . . . 14
loglik . . . 16
nb . . . 17
plot.bnc_dag . . . 18
predict.bnc_fit . . . 18
tan_chowliu . . . 19
voting . . . 20

Index **21**

`accuracy` *Compute predictive accuracy.*

Description

Compute predictive accuracy.

Usage

```
accuracy(x, y)
```

Arguments

`x`	A vector of predicted labels.
`y`	A vector of true labels.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
p <- predict(nb, car)
accuracy(p, car$class)
```

aode *Learn an AODE ensemble.*

Description

If there is a single predictor then returns a naive Bayes.

Usage

```
aode(class, dataset, features = NULL)
```

Arguments

class	A character. Name of the class variable.
dataset	The data frame from which to learn the classifier.
features	A character vector. The names of the features. This argument is ignored if dataset is provided.

Value

A bnc_aode or a bnc_dag (if returning a naive Bayes)

as_mlr *Convert to* mlr.

Description

Convert a bnc_bn to a Learner object.

Usage

```
as_mlr(x, dag, id = "1")
```

Arguments

x	A bnc_bn object.
dag	A logical. Whether to learn structure on each training subsample. Parameters are always learned.
id	A character.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
## Not run: library(mlr)
## Not run: nb_mlr <- as_mlr(nb, dag = FALSE, id = "ode_cl_aic")
## Not run: nb_mlr
```

bnc *Learn network structure and parameters.*

Description

A convenience function to learn the structure and parameters in a single call. Must provide the name of the structure learning algorithm function; see `bnclassify` for the list.

Usage

```
bnc(
  dag_learner,
  class,
  dataset,
  smooth,
  dag_args = NULL,
  awnb_trees = NULL,
  awnb_bootstrap = NULL,
  manb_prior = NULL,
  wanbia = NULL
)
```

Arguments

`dag_learner`	A character. Name of the structure learning function.
`class`	A character. Name of the class variable.
`dataset`	The data frame from which to learn network structure and parameters.
`smooth`	A numeric. The smoothing value (α) for Bayesian parameter estimation. Non-negative.
`dag_args`	A list. Optional additional arguments to `dag_learner`.
`awnb_trees`	An integer. The number (M) of bootstrap samples to generate.
`awnb_bootstrap`	A numeric. The size of the bootstrap subsample, relative to the size of `dataset` (given in [0,1]).
`manb_prior`	A numeric. The prior probability for an arc between the class and any feature.
`wanbia`	A logical. If `TRUE`, WANBIA feature weighting is performed.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
nb_manb <- bnc('nb', 'class', car, smooth = 1, manb_prior = 0.3)
ode_cl_aic <- bnc('tan_cl', 'class', car, smooth = 1, dag_args = list(score = 'aic'))
```

bnc_bn *Bayesian network classifier with structure and parameters.*

Description

A Bayesian network classifier with structure and parameters. Returned by `lp` and `bnc` functions. You can use it to classify data (with `predict`). Can estimate its predictive accuracy with `cv`, plot its structure (with `plot`), print a summary to console (`print`), inspect it with functions documented in `inspect_bnc_bn` and `inspect_bnc_dag`, and convert it to mlr, grain, and graph objects –see `as_mlr` and `grain_and_graph`.

Examples

```
data(car)
tan <- bnc('tan_cl', 'class', car, smooth = 1)
tan
p <- predict(tan, car)
head(p)
## Not run: plot(tan)
nparams(tan)
```

bnc_dag *Bayesian network classifier structure.*

Description

A Bayesian network classifier structure, returned by functions such as `nb` and `tan_cl`. You can plot its structure (with `plot`), print a summary to console (`print`), inspect it with functions documented in `inspect_bnc_dag`, and convert it to a graph object with `grain_and_graph`.

Examples

```
data(car)
nb <- tan_cl('class', car)
nb
## Not run: plot(nb)
narcs(nb)
```

`bnclassify` *Learn discrete Bayesian network classifiers from data.*

Description

State-of-the-art algorithms for learning discrete Bayesian network classifiers from data, with functions prediction, model evaluation and inspection.

Details

The learn more about the package, start with the vignettes: `browseVignettes(package = "bnclassify")`. The following is a list of available functionalities:

Structure learning algorithms:

- `nb`: Naive Bayes (Minsky, 1961)
- `tan_cl`: Chow-Liu's algorithm for one-dependence estimators (CL-ODE) (Friedman et al., 1997)
- `fssj`: Forward sequential selection and joining (FSSJ) (Pazzani, 1996)
- `bsej`: Backward sequential elimination and joining (BSEJ) (Pazzani, 1996)
- `tan_hc`: Hill-climbing tree augmented naive Bayes (TAN-HC) (Keogh and Pazzani, 2002)
- `tan_hcsp`: Hill-climbing super-parent tree augmented naive Bayes (TAN-HCSP) (Keogh and Pazzani, 2002)
- `aode`: Averaged one-dependence estimators (AODE) (Webb et al., 2005)

Parameter learning methods (`lp`):

- Bayesian and maximum likelihood estimation
- Weighting attributes to alleviate naive bayes' independence assumption (WANBIA) (Zaidi et al., 2013)
- Attribute-weighted naive Bayes (AWNB) (Hall, 2007)
- Model averaged naive Bayes (MANB) (Dash and Cooper, 2002)

Model evaluating:

- `cv`: Cross-validated estimate of accuracy
- `logLik`: Log-likelihood
- `AIC`: Akaike's information criterion (AIC)
- `BIC`: Bayesian information criterion (BIC)

Predicting:

- `predict`: Inference for complete and/or incomplete data (the latter through `gRain`)

Inspecting models:

- plot: Structure plotting (through igraph)
- print: Summary
- params: Access conditional probability tables
- nparams: Number of free parameters
- and more. See inspect_bnc_dag and inspect_bnc_bn.

Author(s)

Maintainer: Mihaljevic Bojan <boki.mihaljevic@gmail.com> [copyright holder]

Authors:

- Bielza Concha <mcbielza@fi.upm.es>
- Larranaga Pedro <pedro.larranaga@fi.upm.es>

Other contributors:

- Wickham Hadley (some code extracted from memoise package) [contributor]

References

Bielza C and Larranaga P (2014), Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, **47**(1), Article 5.

Dash D and Cooper GF (2002). Exact model averaging with naive Bayesian classifiers. *19th International Conference on Machine Learning (ICML-2002)*, 91-98.

Friedman N, Geiger D and Goldszmidt M (1997). Bayesian network classifiers. *Machine Learning*, **29**, pp. 131–163.

Zaidi NA, Cerquides J, Carman MJ, and Webb GI (2013) Alleviating naive Bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, **14** pp. 1947–1988.

GI. Webb, JR Boughton, and Z Wang (2005) Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, **58**(1) pp. 5–24.

Hall M (2007). A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-Based Systems*, **20**(2), pp. 120-126.

Koegh E and Pazzani M (2002).Learning the structure of augmented Bayesian classifiers. In *International Journal on Artificial Intelligence Tools*, **11**(4), pp. 587-601.

Koller D, Friedman N (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.

Pazzani M (1996). Constructive induction of Cartesian product attributes. In *Proceedings of the Information, Statistics and Induction in Science Conference (ISIS-1996)*, pp. 66-77

See Also

Useful links:

- https://github.com/bmihaljevic/bnclassify
- Report bugs at https://github.com/bmihaljevic/bnclassify/issues

---

car *Car Evaluation Data Set.*

Description

Data set from the UCI repository: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

Format

A data.frame with 7 columns and 1728 rows.

Source

https://goo.gl/GTXrCz

---

cmi *Compute the (conditional) mutual information between two variables.*

Description

Computes the (conditional) mutual information between two variables. If z is not NULL then returns the conditional mutual information, $I(X;Y|Z)$. Otherwise, returns mutual information, $I(X;Y)$.

Usage

```
cmi(x, y, dataset, z = NULL, unit = "log")
```

Arguments

x	A length one character.
y	A length one character.
dataset	A data frame. Must contain x, y and, optionally, z columns.
z	A character vector.
unit	A character. Logarithm base. See entropy package.

Details

$I(X;Y|Z) = H(X|Z) + H(Y|Z) - H(X,Y,Z) - H(Z)$, where $H()$ is Shannon's entropy.

Examples

```
data(car)
cmi('maint', 'class', car)
```

cv *Estimate predictive accuracy with stratified cross validation.*

Description

Estimate predictive accuracy of a classifier with stratified cross validation. It learns the models from the training subsamples by repeating the learning procedures used to obtain `x`. It can keep the network structure fixed and re-learn only the parameters, or re-learn both structure and parameters.

Usage

```
cv(x, dataset, k, dag = TRUE, mean = TRUE)
```

Arguments

`x`	List of `bnc_bn` or a single `bnc_bn`. The classifiers to evaluate.
`dataset`	The data frame on which to evaluate the classifiers.
`k`	An integer. The number of folds.
`dag`	A logical. Whether to learn structure on each training subsample. Parameters are always learned.
`mean`	A logical. Whether to return mean accuracy for each classifier or to return a k-row matrix with accuracies per fold.

Value

A numeric vector of same length as `x`, giving the predictive accuracy of each classifier. If `mean = FALSE` then a matrix with k rows and a column per each classifier in `x`.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
# CV a single classifier
cv(nb, car, k = 10)
nb_manb <- bnc('nb', 'class', car, smooth = 1, manb_prior = 0.5)
cv(list(nb=nb, manb=nb_manb), car, k = 10)
# Get accuracies on each fold
cv(list(nb=nb, manb=nb_manb), car, k = 10, mean = FALSE)
ode <- bnc('tan_cl', 'class', car, smooth = 1, dag_args = list(score = 'aic'))
# keep structure fixed accross training subsamples
cv(ode, car, k = 10, dag = FALSE)
```

grain_and_graph *Convert to igraph and gRain.*

Description

Convert a bnc_dag to igraph and grain objects.

Usage

```
as_igraph(x)

as_grain(x)
```

Arguments

x	The bnc_bn object. The Bayesian network classifier.

Functions

- as_igraph(): Convert to a graphNEL.
- as_grain(): Convert to a grain.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
# Requires the grain and igraph packages installed
## Not run: g <- as_grain(nb)
## Not run: gRain::querygrain.grain(g)$buying
```

greedy_wrapper *Learn Bayesian network classifiers in a a greedy wrapper fashion.*

Description

Greedy wrapper algorithms for learning Bayesian network classifiers. All algorithms use cross-validated estimate of predictive accuracy to evaluate candidate structures.

Usage

```
fssj(class, dataset, k, epsilon = 0.01, smooth = 0, cache_reset = NULL)

bsej(class, dataset, k, epsilon = 0.01, smooth = 0, cache_reset = NULL)

tan_hc(class, dataset, k, epsilon = 0.01, smooth = 0, cache_reset = NULL)

kdb(
  class,
  dataset,
  k,
  kdbk = 2,
  epsilon = 0.01,
  smooth = 0,
  cache_reset = NULL
)

tan_hcsp(class, dataset, k, epsilon = 0.01, smooth = 0, cache_reset = NULL)
```

Arguments

class	A character. Name of the class variable.
dataset	The data frame from which to learn the classifier.
k	An integer. The number of folds.
epsilon	A numeric. Minimum absolute improvement in accuracy required to keep searching.
smooth	A numeric. The smoothing value (α) for Bayesian parameter estimation. Non-negative.
cache_reset	A numeric. Number of iterations after which to reset the cache of conditional probability tables. A small number reduces the amount of memory used. NULL means the cache is never reset (the default).
kdbk	An integer. The maximum number of feature parents per feature.

Value

A bnc_dag object.

References

Pazzani M (1996). Constructive induction of Cartesian product attributes. In *Proceedings of the Information, Statistics and Induction in Science Conference (ISIS-1996)*, pp. 66-77

Koegh E and Pazzani M (2002).Learning the structure of augmented Bayesian classifiers. In *International Journal on Artificial Intelligence Tools*, **11**(4), pp. 587-601.

Examples

```
data(car)
tanhc <- tan_hc('class', car, k = 5, epsilon = 0)
## Not run: plot(tanhc)
```

`inspect_bnc_bn` *Inspect a Bayesian network classifier (with structure and parameters).*

Description

Functions for inspecting a `bnc_bn` object. In addition, you can query this object with the functions documented in `inspect_bnc_dag`.

Usage

```
nparams(x)

manb_arc_posterior(x)

awnb_weights(x)

params(x)

values(x)

classes(x)
```

Arguments

`x`	The `bnc_bn` object. The Bayesian network classifier.

Functions

- `nparams()`: Returns the number of free parameters in the model.
- `manb_arc_posterior()`: Returns the posterior of each arc from the class according to the MANB method.
- `awnb_weights()`: Returns the AWNB feature weights.
- `params()`: Returns the list of CPTs, in the same order as `vars`.
- `values()`: Returns the possible values of each variable, in the same order as `vars`.
- `classes()`: Returns the possible values of the class variable.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
nparams(nb)
nb <- bnc('nb', 'class', car, smooth = 1, manb_prior = 0.5)
manb_arc_posterior(nb)
nb <- bnc('nb', 'class', car, smooth = 1, awnb_bootstrap = 0.5)
awnb_weights(nb)
```

`inspect_bnc_dag` *Inspect a Bayesian network classifier structure.*

Description

Functions for inspecting a bnc_dag object.

Usage

```
class_var(x)

features(x)

vars(x)

families(x)

modelstring(x)

feature_families(x)

narcs(x)

is_semi_naive(x)

is_anb(x)

is_nb(x)

is_ode(x)
```

Arguments

x	The bnc_dag object. The Bayesian network classifier structure.

Functions

- `class_var()`: Returns the class variable.
- `features()`: Returns the features.
- `vars()`: Returns all variables (i.e., features + class).
- `families()`: Returns the family of each variable.
- `modelstring()`: Returns the model string of the network in bnlearn format (adding a space in between two families).
- `feature_families()`: Returns the family of each feature.
- `narcs()`: Returns the number of arcs.
- `is_semi_naive()`: Returns TRUE if `x` is a semi-naive Bayes.
- `is_anb()`: Returns TRUE if `x` is an augmented naive Bayes.
- `is_nb()`: Returns TRUE if `x` is a naive Bayes.
- `is_ode()`: Returns TRUE if `x` is a one-dependence estimator.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
narcs(nb)
is_ode(nb)
```

---

learn_params *Learn the parameters of a Bayesian network structure.*

Description

Learn parameters with maximum likelihood or Bayesian estimation, the weighting attributes to alleviate naive bayes' independence assumption (WANBIA), attribute weighted naive Bayes (AWNB), or the model averaged naive Bayes (MANB) methods. Returns a bnc_bn.

Usage

```
lp(
  x,
  dataset,
  smooth,
  awnb_trees = NULL,
  awnb_bootstrap = NULL,
  manb_prior = NULL,
  wanbia = NULL
)
```

Arguments

x	The bnc_dag object. The Bayesian network classifier structure.
dataset	The data frame from which to learn network parameters.
smooth	A numeric. The smoothing value (α) for Bayesian parameter estimation. Non-negative.
awnb_trees	An integer. The number (M) of bootstrap samples to generate.
awnb_bootstrap	A numeric. The size of the bootstrap subsample, relative to the size of dataset (given in [0,1]).
manb_prior	A numeric. The prior probability for an arc between the class and any feature.
wanbia	A logical. If TRUE, WANBIA feature weighting is performed.

Details

lp learns the parameters of each local distribution $\theta_{ijk} = P(X_i = k \mid \mathbf{Pa}(X_i) = j)$ as

$$\theta_{ijk} = \frac{N_{ijk} + \alpha}{N_{ij.} + r_i \alpha},$$

where N_{ijk} is the number of instances in dataset in which $X_i = k$ and $\mathbf{Pa}(X_i) = j$, $N_{ij.} = \sum_{k=1}^{r_i} N_{ijk}$, r_i is the cardinality of X_i, and all hyperparameters of the Dirichlet prior equal to α. $\alpha = 0$ corresponds to maximum likelihood estimation. Returns a uniform distribution when $N_{ij.} + r_i \alpha = 0$. With partially observed data, the above amounts to *available case analysis*.

WANBIA learns a unique exponent 'weight' per feature. They are computed by optimizing conditional log-likelihood, and are bounded with all $w_i \in [0, 1]$. For WANBIA estimates, set wanbia to TRUE.

In order to get the AWNB parameter estimate, provide either the awnb_bootstrap and/or the awnb_trees argument. The estimate is:

$$\theta_{ijk}^{AWNB} = \frac{\theta_{ijk}^{w_i}}{\sum_{k=1}^{r_i} \theta_{ijk}^{w_i}},$$

while the weights w_i are computed as

$$w_i = \frac{1}{M} \sum_{t=1}^{M} \sqrt{\frac{1}{d_{ti}}},$$

where M is the number of bootstrap samples from dataset and d_{ti} the minimum testing depth of X_i in an unpruned classification tree learned from the t-th subsample ($d_{ti} = 0$ if X_i is omitted from t-th tree).

The MANB parameters correspond to Bayesian model averaging over the naive Bayes models obtained from all 2^n subsets over the n features. To get MANB parameters, provide the manb_prior argument.

Value

A bnc_bn object.

References

Hall M (2004). A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-based Systems*, **20**(2), 120-126.

Dash D and Cooper GF (2002). Exact model averaging with naive Bayesian classifiers. *19th International Conference on Machine Learning (ICML-2002)*, 91-98.

Pigott T D (2001) A review of methods for missing data. *Educational research and evaluation*, **7**(4), 353-383.

Examples

```
data(car)
nb <- nb('class', car)
# Maximum likelihood estimation
mle <- lp(nb, car, smooth = 0)
# Bayesian estimaion
bayes <- lp(nb, car, smooth = 0.5)
# MANB
manb <- lp(nb, car, smooth = 0.5, manb_prior = 0.5)
# AWNB
awnb <- lp(nb, car, smooth = 0.5, awnb_trees = 10)
```

---

loglik *Compute (penalized) log-likelihood.*

Description

Compute (penalized) log-likelihood and conditional log-likelihood score of a bnc_bn object on a data set. Requires a data frame argument in addition to `object`.

Usage

```
## S3 method for class 'bnc_bn'
AIC(object, ...)

## S3 method for class 'bnc_bn'
BIC(object, ...)

## S3 method for class 'bnc_bn'
logLik(object, ...)

cLogLik(object, ...)
```

Arguments

`object`	A bnc_bn object.
`...`	A data frame ($\mathcal{D}$).

Details

log-likelihood = $logP(\mathcal{D} \mid \theta)$,

Akaike's information criterion (AIC) = $logP(\mathcal{D} \mid \theta) - \frac{1}{2}|\theta|$,

The Bayesian information criterion (BIC) score: = $logP(\mathcal{D} \mid \theta) - \frac{\log N}{2}|\theta|$,

where $|\theta|$ is the number of free parameters in `object`, $\mathcal{D}$ is the data set and N is the number of instances in $\mathcal{D}$.

`cLogLik` computes the conditional log-likelihood of the model.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
logLik(nb, car)
AIC(nb, car)
BIC(nb, car)
cLogLik(nb, car)
```

---

nb *Learn a naive Bayes network structure.*

Description

Learn a naive Bayes network structure.

Usage

```
nb(class, dataset = NULL, features = NULL)
```

Arguments

`class`	A character. Name of the class variable.
`dataset`	The data frame from which to learn the classifier.
`features`	A character vector. The names of the features. This argument is ignored if `dataset` is provided.

Value

A bnc_dag object.

Examples

```
data(car)
nb <- nb('class', car)
nb2 <- nb('class', features = letters[1:10])
## Not run: plot(nb2)
```

plot.bnc_dag *Plot the structure.*

Description

If node labels are to small to be viewed properly, you may fix label fontsize with argument fontsize. Also, you may try multiple different layouts.

Usage

```
## S3 method for class 'bnc_dag'
plot(x, y, layoutType = "dot", fontsize = NULL, ...)
```

Arguments

x	The bnc_dag object. The Bayesian network classifier structure.
y	Not used
layoutType	a character. Optional.
fontsize	integer Font size for node labels. Optional.
...	Not used.

Examples

```
# Requires the igraph package to be installed.
data(car)
nb <- nb('class', car)
nb <- nb('class', car)
## Not run: plot(nb)
## Not run: plot(nb, fontsize = 20)
## Not run: plot(nb, layoutType = 'circo')
## Not run: plot(nb, layoutType = 'fdp')
## Not run: plot(nb, layoutType = 'osage')
## Not run: plot(nb, layoutType = 'twopi')
## Not run: plot(nb, layoutType = 'neato')
```

predict.bnc_fit *Predicts class labels or class posterior probability distributions.*

Description

Predicts class labels or class posterior probability distributions.

Usage

```
## S3 method for class 'bnc_fit'
predict(object, newdata, prob = FALSE, ...)
```

Arguments

object	A bnc_bn object.
newdata	A data frame containing observations whose class has to be predicted.
prob	A logical. Whether class posterior probability should be returned.
...	Ignored.

Details

Ties are resolved randomly. Inference is much slower if newdata contains NAs.

Value

If prob=FALSE, then returns a length-N factor with the same levels as the class variable in x, where N is the number of rows in newdata. Each element is the most likely class for the corresponding row in newdata. If prob=TRUE, returns a N by C numeric matrix, where C is the number of classes; each row corresponds to the class posterior of the instance.

Examples

```
data(car)
nb <- bnc('nb', 'class', car, smooth = 1)
p <- predict(nb, car)
head(p)
p <- predict(nb, car, prob = TRUE)
head(p)
```

---

tan_chowliu *Learns a one-dependence estimator using Chow-Liu's algorithm.*

Description

Learns a one-dependence Bayesian classifier using Chow-Liu's algorithm, by maximizing either log-likelihood, the AIC or BIC scores; maximizing log-likelihood corresponds to the well-known tree augmented naive Bayes (Friedman et al., 1997). When maximizing AIC or BIC the output might be a forest-augmented rather than a tree-augmented naive Bayes.

Usage

```
tan_cl(class, dataset, score = "loglik", root = NULL)
```

Arguments

class	A character. Name of the class variable.
dataset	The data frame from which to learn the classifier.
score	A character. The score to be maximized. 'loglik', 'bic', and 'aic' return the maximum likelihood, maximum BIC and maximum AIC tree/forest, respectively.
root	A character. The feature to be used as root of the augmenting tree. Only one feature can be supplied, even in case of an augmenting forest. This argument is optional.

Value

A bnc_dag object.

References

Friedman N, Geiger D and Goldszmidt M (1997). Bayesian network classifiers. *Machine Learning*, **29**, pp. 131–163.

Examples

```
data(car)
ll <- tan_cl('class', car, score = 'loglik')
## Not run: plot(ll)
ll <- tan_cl('class', car, score = 'loglik', root = 'maint')
## Not run: plot(ll)
aic <- tan_cl('class', car, score = 'aic')
bic <- tan_cl('class', car, score = 'bic')
```

---

voting *Congress Voting Data Set.*

Description

Data set from the UCI repository https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records.

Format

A data.frame with 17 columns and 435 rows.

Source

https://goo.gl/GTXrCz

Index

accuracy, 2
AIC, *6*
AIC.bnc_bn *(*loglik*)*, 16
aode, 3, *6*
as_grain *(*grain_and_graph*)*, 10
as_igraph *(*grain_and_graph*)*, 10
as_mlr, 3, *5*
awnb_weights *(*inspect_bnc_bn*)*, 12

BIC, *6*
BIC.bnc_bn *(*loglik*)*, 16
bnc, 4, *5*
bnc_bn, *3*, 5, *9, 10*, *12*, *14–16*, *19*
bnc_dag, 5, *10, 11*, *13*, *15*, *17, 18*, *20*
bnclassify, *4*, 6
bnclassify-package *(*bnclassify*)*, 6
bsej, *6*
bsej *(*greedy_wrapper*)*, 10

car, 8
class_var *(*inspect_bnc_dag*)*, 13
classes *(*inspect_bnc_bn*)*, 12
cLogLik *(*loglik*)*, 16
cmi, 8
cv, *5, 6*, 9

families *(*inspect_bnc_dag*)*, 13
feature_families *(*inspect_bnc_dag*)*, 13
features *(*inspect_bnc_dag*)*, 13
fssj, *6*
fssj *(*greedy_wrapper*)*, 10

grain, *10*
grain_and_graph, *5*, 10
greedy_wrapper, 10

inspect_bnc_bn, *5*, *7*, 12
inspect_bnc_dag, *5*, *7*, *12*, 13
is_anb *(*inspect_bnc_dag*)*, 13
is_nb *(*inspect_bnc_dag*)*, 13
is_ode *(*inspect_bnc_dag*)*, 13
is_semi_naive *(*inspect_bnc_dag*)*, 13

kdb *(*greedy_wrapper*)*, 10

learn_params, 14
Learner, *3*
logLik, *6*
loglik, 16
logLik.bnc_bn *(*loglik*)*, 16
lp, *5, 6*
lp *(*learn_params*)*, 14

manb_arc_posterior *(*inspect_bnc_bn*)*, 12
modelstring *(*inspect_bnc_dag*)*, 13

narcs *(*inspect_bnc_dag*)*, 13
nb, *5, 6*, 17
nparams, *7*
nparams *(*inspect_bnc_bn*)*, 12

params, *7*
params *(*inspect_bnc_bn*)*, 12
plot, *5*, *7*
plot.bnc_dag, 18
predict, *5, 6*
predict.bnc_fit, 18
print, *5*, *7*

tan_chowliu, 19
tan_cl, *5, 6*
tan_cl *(*tan_chowliu*)*, 19
tan_hc, *6*
tan_hc *(*greedy_wrapper*)*, 10
tan_hcsp, *6*
tan_hcsp *(*greedy_wrapper*)*, 10

values *(*inspect_bnc_bn*)*, 12
vars, *12*
vars *(*inspect_bnc_dag*)*, 13
voting, 20