
Bayes Documentation

Release 0.1.0

Krzysztof Joachimiak

January 14, 2017

1	Bayes API	3
1.1	Classifiers	3
1.1.1	Complement Naive Bayes	3
1.1.2	Locally Weighted Naive Bayes	6
1.1.3	Negation Naive Bayes	7
1.1.4	Selective Naive Bayes	10
1.1.5	Universal-set Naive Bayes	12
1.2	Utils	15
1.2.1	Submodules	15
1.2.2	bayes.utils.math_utils	15
1.2.3	bayes.utils.doc_utils	15
2	Algorithms	17
3	Installation	19
4	Usage	21
	Python Module Index	23

This module contains several variants of well-known Naive Bayes Classifier.

Contents:

1.1 Classifiers

1.1.1 Complement Naive Bayes

class `bayes.classifiers.cnbc.ComplementNB` (*alpha=1.0, weight_normalized=False*)

Bases: `bayes.base.BaseNB`

Complement Naive Bayes classifier

Parameters

- **alpha** (*float*) – Smoothing parameter
- **weight_normalized** (*bool, default False*) – Enable Weight-normalized Complement Naive Bayes method.

alpha_sum_

int

Sum of alpha params

classes_

array, shape (n_classes,)

Classes list

class_count_

array, shape (n_classes,)

number of training samples observed in each class.

Examples

```
>>> from sklearn.datasets import fetch_20newsgroups
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> from bayes.classifiers import ComplementNB
Prepare data
>>> vectorizer = CountVectorizer()
>>> categories = ['alt.atheism', 'talk.religion.misc', 'comp.graphics', 'sci.space']
Train set
>>> newsgroups_train = fetch_20newsgroups(subset='train', categories=categories, shuffle=True)
>>> train_vectors = vectorizer.fit_transform(newsgroups_train.data)
Test set
```

```
>>> newsgroups_test = fetch_20newsgroups(subset='test', categories=categories, shuffle=True)
>>> test_vectors = vectorizer.transform(newsgroups_test.data)
>>> clf = ComplementNB()
>>> clf.fit(newsgroups_train, train_vectors).accuracy_score(newsgroups_test, test_vectors)
```

References

Rennie J. D. M., Shih L., Teevan J., Karger D. R. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers

<https://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf>

`__delattr__`

`x.__delattr__('name') <==> del x.name`

`__format__` ()

default object formatter

`__getattr__`

`x.__getattr__('name') <==> x.name`

`__hash__`

`__reduce__` ()

helper for pickle

`__reduce_ex__` ()

helper for pickle

`__setattr__`

`x.__setattr__('name', value) <==> x.name = value`

`__sizeof__` () → int

size of object in memory, in bytes

`__str__`

`accuracy_score` (*X*, *y*)

Return accuracy score

Parameters

- **X** (*{array-like, sparse matrix}*, *shape = [n_samples, n_features]*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape = [n_samples]*) – Target values.

Returns `accuracy_score` – Accuracy on the given test set

Return type float

`class_log_proba_`

Log probability of class occurrence

`complement_class_count_`

Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class *c*

`complement_class_log_proba_`

Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class *c*

fit (*X*, *y*)

Fit model to given training set

Parameters

- **x** (*array-like, shape (n_samples, n_features)*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like, shape (n_samples,)*) – Target values.

Returns self – Returns self.

Return type Naive Bayes estimator object

get_params (*deep=True*)

Get parameters for this estimator.

Parameters deep (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns params – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit (*X*, *y*, *classes=None*)

Incremental fit on a batch of samples.

Parameters

- **x** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like, shape = [n_samples]*) – Target values.
- **classes** (*array-like, shape = [n_classes], optional (default=None)*) – List of all the classes that can possibly appear in the *y* vector. Must be provided at the first call to `partial_fit`, can be omitted in subsequent calls.

Returns self – Returns self.

Return type object

predict (*X*)

Perform classification on an array of test vectors *X*.

Parameters X (*array-like, shape = [n_samples, n_features]*) – Unseen samples vector

Returns C – Predicted target values for *X*

Return type array, shape = [n_samples]

predict_log_proba (*X*)

Return log-probability estimates for the test vector *X*.

Parameters X (*array-like, shape = [n_samples, n_features]*) –

Returns C – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute `classes_`.

Return type array-like, shape = [n_samples, n_classes]

predict_proba (*X*)

Return probability estimates for the test vector *X*. :param *X*: :type *X*: array-like, shape = [n_samples, n_features]

Returns *C* – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

safe_matmult (*input_array*, *internal_array*)

set_params (***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form <component>__<parameter> so that it's possible to update each component of a nested object.

Returns

Return type self

1.1.2 Locally Weighted Naive Bayes

class bayes.classifiers.lwnb.**LocallyWeightedNB** (*alpha=1.0*, *weight_normalized=False*)

Bases: bayes.base.BaseNB

Locally Weighted Naive Bayes classifier

Parameters

- **alpha** (*float*) – Smoothing parameter
- **weight_normalized** (*bool*, *default False*) – Enable Weight-normalized Complement Naive Bayes method.

References

Frank E., Hall M., Pfahringer B. (2003).

http://www.cs.waikato.ac.nz/~eibe/pubs/UAI_200.pdf

__delattr__

x.__delattr__('name') <==> del *x*.name

__format__ ()

default object formatter

__getattr__

x.__getattr__('name') <==> *x*.name

__hash__

__reduce__ ()

helper for pickle

__reduce_ex__ ()

helper for pickle

__setattr__

x.__setattr__('name', value) <==> *x*.name = value

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`

`accuracy_score(X, y)`
Return accuracy score

Parameters

- **X** (*{array-like, sparse matrix}*, *shape = [n_samples, n_features]*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape = [n_samples]*) – Target values.

Returns `accuracy_score` – Accuracy on the given test set

Return type float

`class_log_proba_`
Log probability of class occurrence

`complement_class_count_`
Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class *c*

`complement_class_log_proba_`
Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class *c*

`fit(X, y)`

`get_params()`

`partial_fit(X, y, classes=None)`

`predict(X)`

`predict_log_proba(X)`

`predict_proba(X)`

`safe_matmult(input_array, internal_array)`

`set_params(**params)`

1.1.3 Negation Naive Bayes

`class bayes.classifiers.nnb.NegationNB(alpha=1.0)`
Bases: `bayes.base.BaseNB`

Negation Naive Bayes classifier

Parameters `alpha` (*float*) – Smoothing parameter

References

Komiya K., Sato N., Fujimoto K., Kotani Y. (2011). Negation Naive Bayes for Categorization of Product Pages on the Web

<http://www.aclweb.org/anthology/R11-1083.pdf>

__delattr__
 x.__delattr__('name') <==> del x.name

__format__ ()
 default object formatter

__getattr__
 x.__getattr__('name') <==> x.name

__hash__

__reduce__ ()
 helper for pickle

__reduce_ex__ ()
 helper for pickle

__setattr__
 x.__setattr__('name', value) <==> x.name = value

__sizeof__ () → int
 size of object in memory, in bytes

__str__

accuracy_score (X, y)
 Return accuracy score

Parameters

- **X** (*array-like, sparse matrix*), *shape* = [*n_samples*, *n_features*] – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape* = [*n_samples*]) – Target values.

Returns **accuracy_score** – Accuracy on the given test set

Return type float

class_log_proba
 Log probability of class occurrence

complement_class_count
 Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class c

complement_class_log_proba
 Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class c

fit (X, y)
 Fit model to given training set

Parameters

- **X** (*array-like*, *shape* (*n_samples*, *n_features*)) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape* (*n_samples*,)) – Target values.

Returns **self** – Returns self.

Return type Naive Bayes estimator object

get_params (*deep=True*)

Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit (*X, y, classes=None*)

Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like, shape = [n_samples]*) – Target values.
- **classes** (*array-like, shape = [n_classes], optional (default=None)*) – List of all the classes that can possibly appear in the *y* vector. Must be provided at the first call to *partial_fit*, can be omitted in subsequent calls.

Returns **self** – Returns self.

Return type object

predict (*X*)

Perform classification on an array of test vectors *X*.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) – Unseen samples vector

Returns **C** – Predicted target values for *X*

Return type array, shape = [n_samples]

predict_log_proba (*X*)

Return log-probability estimates for the test vector *X*.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) –

Returns **C** – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

predict_proba (*X*)

Return probability estimates for the test vector *X*. :param X: :type X: array-like, shape = [n_samples, n_features]

Returns **C** – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

safe_matmult (*input_array, internal_array*)

set_params (***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns**Return type** `self`

1.1.4 Selective Naive Bayes

class `bayes.classifiers.snb.SelectiveNB` (*alpha=1.0*)Bases: `bayes.base.BaseNB`

Selective Naive Bayes classifier

Parameters `alpha` (*float*) – Smoothing parameter**References**

Komiya K., Ito Y., Kotani Y. (2013). New Naive Bayes Methods using Data from All Classes

<http://aia-i.com/ijai/sample/vol5/no1/1-13.pdf>`__delattr__``x.__delattr__('name') <==> del x.name``__format__` ()

default object formatter

`__getattr__``x.__getattr__('name') <==> x.name``__hash__``__reduce__` ()

helper for pickle

`__reduce_ex__` ()

helper for pickle

`__setattr__``x.__setattr__('name', value) <==> x.name = value``__sizeof__` () → int

size of object in memory, in bytes

`__str__`**accuracy_score** (*X, y*)

Return accuracy score

Parameters

- **X** (*{array-like, sparse matrix}*, *shape = [n_samples, n_features]*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape = [n_samples]*) – Target values.

Returns `accuracy_score` – Accuracy on the given test set**Return type** `float`

class_log_proba_

Log probability of class occurrence

complement_class_count_

Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class *c*

complement_class_log_proba_

Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class *c*

fit (*X*, *y*)

Fit model to given training set

Parameters

- **X** (*array-like*, *shape* (*n_samples*, *n_features*)) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape* (*n_samples*,)) – Target values.

Returns self – Returns self.

Return type Naive Bayes estimator object

get_params (*deep=True*)

Get parameters for this estimator.

Parameters **deep** (*boolean*, *optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns params – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit (*X*, *y*, *classes=None*)

Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}*, *shape* = [*n_samples*, *n_features*]) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape* = [*n_samples*]) – Target values.
- **classes** (*array-like*, *shape* = [*n_classes*], *optional* (*default=None*)) – List of all the classes that can possibly appear in the *y* vector. Must be provided at the first call to `partial_fit`, can be omitted in subsequent calls.

Returns self – Returns self.

Return type object

predict (*X*)

Perform classification on an array of test vectors *X*.

Parameters **X** (*array-like*, *shape* = [*n_samples*, *n_features*]) – Unseen samples vector

Returns C – Predicted target values for *X*

Return type array, *shape* = [*n_samples*]

predict_log_proba (*X*)

Return log-probability estimates for the test vector *X*.

Parameters *X* (*array-like*, *shape* = [*n_samples*, *n_features*]) –

Returns *C* – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, *shape* = [*n_samples*, *n_classes*]

predict_proba (*X*)

Return probability estimates for the test vector *X*. :param *X*: :type *X*: array-like, *shape* = [*n_samples*, *n_features*]

Returns *C* – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, *shape* = [*n_samples*, *n_classes*]

safe_matmult (*input_array*, *internal_array*)

set_params (***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form <component>__<parameter> so that it's possible to update each component of a nested object.

Returns

Return type self

1.1.5 Universal-set Naive Bayes

class bayes.classifiers.unb.**UniversalSetNB** (*alpha*=1.0)

Bases: bayes.base.BaseNB

Universal-set Naive Bayes classifier

Parameters *alpha* (*float*) – Smoothing parameter

References

Komiya K., Ito Y., Kotani Y. (2013). New Naive Bayes Methods using Data from All Classes

<http://aia-i.com/ijai/sample/vol5/no1/1-13.pdf>

__delattr__

x.__delattr__(*'name'*) <==> del *x.name*

__format__ ()

default object formatter

__getattr__

x.__getattr__(*'name'*) <==> *x.name*

__hash__

__reduce__ ()

helper for pickle

`__reduce_ex__()`

helper for pickle

`__setattr__`

`x.__setattr__('name', value) <==> x.name = value`

`__sizeof__()` → int

size of object in memory, in bytes

`__str__`

accuracy_score (*X*, *y*)

Return accuracy score

Parameters

- **X** (*array-like, sparse matrix*), *shape* = [*n_samples*, *n_features*]) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape* = [*n_samples*]) – Target values.

Returns **accuracy_score** – Accuracy on the given test set

Return type float

class_log_proba

Log probability of class occurrence

complement_class_count

Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class *c*

complement_class_log_proba

Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class *c*

fit (*X*, *y*)

Fit model to given training set

Parameters

- **X** (*array-like*, *shape* (*n_samples*, *n_features*)) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape* (*n_samples*,)) – Target values.

Returns **self** – Returns self.

Return type Naive Bayes estimator object

get_params (*deep=True*)

Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit (*X*, *y*, *classes=None*)

Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}*, *shape = [n_samples, n_features]*) – Training vectors, where *n_samples* is the number of samples and *n_features* is the number of features.
- **y** (*array-like*, *shape = [n_samples]*) – Target values.
- **classes** (*array-like*, *shape = [n_classes]*, *optional (default=None)*) – List of all the classes that can possibly appear in the *y* vector. Must be provided at the first call to `partial_fit`, can be omitted in subsequent calls.

Returns self – Returns self.

Return type object

predict (*X*)

Perform classification on an array of test vectors *X*.

Parameters X (*array-like*, *shape = [n_samples, n_features]*) – Unseen samples vector

Returns C – Predicted target values for *X*

Return type array, *shape = [n_samples]*

predict_log_proba (*X*)

Return log-probability estimates for the test vector *X*.

Parameters X (*array-like*, *shape = [n_samples, n_features]*) –

Returns C – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute `classes_`.

Return type array-like, *shape = [n_samples, n_classes]*

predict_proba (*X*)

Return probability estimates for the test vector *X*. :param X: :type X: array-like, *shape = [n_samples, n_features]*

Returns C – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute `classes_`.

Return type array-like, *shape = [n_samples, n_classes]*

safe_matmult (*input_array*, *internal_array*)

set_params (***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns

Return type self

1.2 Utils

1.2.1 Submodules

1.2.2 bayes.utils.math_utils

1.2.3 bayes.utils.doc_utils

Algorithms

TODO

Installation

You can install this module directly from GitHub repo with command:
or use **pip**

Usage

TODO

- genindex

modindex search

b

`bayes.classifiers.cnb`, 3
`bayes.classifiers.lwnb`, 6
`bayes.classifiers.nnb`, 7
`bayes.classifiers.snb`, 10
`bayes.classifiers.unb`, 12

Symbols

- `__delattr__` (bayes.classifiers.cnb.ComplementNB attribute), 4
- `__delattr__` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 6
- `__delattr__` (bayes.classifiers.nnb.NegationNB attribute), 7
- `__delattr__` (bayes.classifiers.snb.SelectiveNB attribute), 10
- `__delattr__` (bayes.classifiers.unb.UniversalSetNB attribute), 12
- `__format__()` (bayes.classifiers.cnb.ComplementNB method), 4
- `__format__()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 6
- `__format__()` (bayes.classifiers.nnb.NegationNB method), 8
- `__format__()` (bayes.classifiers.snb.SelectiveNB method), 10
- `__format__()` (bayes.classifiers.unb.UniversalSetNB method), 12
- `__getattr__` (bayes.classifiers.cnb.ComplementNB attribute), 4
- `__getattr__` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 6
- `__getattr__` (bayes.classifiers.nnb.NegationNB attribute), 8
- `__getattr__` (bayes.classifiers.snb.SelectiveNB attribute), 10
- `__getattr__` (bayes.classifiers.unb.UniversalSetNB attribute), 12
- `__setattr__` (bayes.classifiers.cnb.ComplementNB attribute), 4
- `__setattr__` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 6
- `__setattr__` (bayes.classifiers.nnb.NegationNB attribute), 8
- `__setattr__` (bayes.classifiers.snb.SelectiveNB attribute), 10
- `__setattr__` (bayes.classifiers.unb.UniversalSetNB attribute), 13
- `__sizeof__()` (bayes.classifiers.cnb.ComplementNB method), 4
- `__sizeof__()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 6
- `__sizeof__()` (bayes.classifiers.nnb.NegationNB method), 8
- `__sizeof__()` (bayes.classifiers.snb.SelectiveNB method), 10
- `__sizeof__()` (bayes.classifiers.unb.UniversalSetNB method), 13
- `__reduce__()` (bayes.classifiers.cnb.ComplementNB method), 4
- `__reduce__()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 6
- `__reduce__()` (bayes.classifiers.nnb.NegationNB method), 8
- `__reduce__()` (bayes.classifiers.snb.SelectiveNB method), 10
- `__reduce__()` (bayes.classifiers.unb.UniversalSetNB method), 12
- `__reduce_ex__()` (bayes.classifiers.cnb.ComplementNB method), 4
- `__reduce_ex__()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 6
- `__reduce_ex__()` (bayes.classifiers.nnb.NegationNB method), 8
- `__reduce_ex__()` (bayes.classifiers.snb.SelectiveNB method), 10
- `__reduce_ex__()` (bayes.classifiers.unb.UniversalSetNB method), 12
- `__setattr__` (bayes.classifiers.cnb.ComplementNB attribute), 4
- `__setattr__` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 6
- `__setattr__` (bayes.classifiers.nnb.NegationNB attribute), 8
- `__setattr__` (bayes.classifiers.snb.SelectiveNB attribute), 10
- `__setattr__` (bayes.classifiers.unb.UniversalSetNB attribute), 13
- `__sizeof__()` (bayes.classifiers.cnb.ComplementNB method), 4
- `__sizeof__()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 6
- `__sizeof__()` (bayes.classifiers.nnb.NegationNB method), 8
- `__sizeof__()` (bayes.classifiers.snb.SelectiveNB method), 10
- `__sizeof__()` (bayes.classifiers.unb.UniversalSetNB method), 13

`__str__` (bayes.classifiers.cnb.ComplementNB attribute), 4
`__str__` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 7
`__str__` (bayes.classifiers.nnb.NegationNB attribute), 8
`__str__` (bayes.classifiers.snb.SelectiveNB attribute), 10
`__str__` (bayes.classifiers.unb.UniversalSetNB attribute), 13

A

`accuracy_score()` (bayes.classifiers.cnb.ComplementNB method), 4
`accuracy_score()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 7
`accuracy_score()` (bayes.classifiers.nnb.NegationNB method), 8
`accuracy_score()` (bayes.classifiers.snb.SelectiveNB method), 10
`accuracy_score()` (bayes.classifiers.unb.UniversalSetNB method), 13
`alpha_sum_` (bayes.classifiers.cnb.ComplementNB attribute), 3

B

bayes.classifiers.cnb (module), 3
 bayes.classifiers.lwnb (module), 6
 bayes.classifiers.nnb (module), 7
 bayes.classifiers.snb (module), 10
 bayes.classifiers.unb (module), 12

C

`class_count_` (bayes.classifiers.cnb.ComplementNB attribute), 3
`class_log_proba_` (bayes.classifiers.cnb.ComplementNB attribute), 4
`class_log_proba_` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 7
`class_log_proba_` (bayes.classifiers.nnb.NegationNB attribute), 8
`class_log_proba_` (bayes.classifiers.snb.SelectiveNB attribute), 10
`class_log_proba_` (bayes.classifiers.unb.UniversalSetNB attribute), 13
`classes_` (bayes.classifiers.cnb.ComplementNB attribute), 3
`complement_class_count_` (bayes.classifiers.cnb.ComplementNB attribute), 4
`complement_class_count_` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 7
`complement_class_count_` (bayes.classifiers.nnb.NegationNB attribute), 8

`complement_class_count_` (bayes.classifiers.snb.SelectiveNB attribute), 11
`complement_class_count_` (bayes.classifiers.unb.UniversalSetNB attribute), 13
`complement_class_log_proba_` (bayes.classifiers.cnb.ComplementNB attribute), 4
`complement_class_log_proba_` (bayes.classifiers.lwnb.LocallyWeightedNB attribute), 7
`complement_class_log_proba_` (bayes.classifiers.nnb.NegationNB attribute), 8
`complement_class_log_proba_` (bayes.classifiers.snb.SelectiveNB attribute), 11
`complement_class_log_proba_` (bayes.classifiers.unb.UniversalSetNB attribute), 13
 ComplementNB (class in bayes.classifiers.cnb), 3

F

`fit()` (bayes.classifiers.cnb.ComplementNB method), 4
`fit()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 7
`fit()` (bayes.classifiers.nnb.NegationNB method), 8
`fit()` (bayes.classifiers.snb.SelectiveNB method), 11
`fit()` (bayes.classifiers.unb.UniversalSetNB method), 13

G

`get_params()` (bayes.classifiers.cnb.ComplementNB method), 5
`get_params()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 7
`get_params()` (bayes.classifiers.nnb.NegationNB method), 8
`get_params()` (bayes.classifiers.snb.SelectiveNB method), 11
`get_params()` (bayes.classifiers.unb.UniversalSetNB method), 13

L

LocallyWeightedNB (class in bayes.classifiers.lwnb), 6

N

NegationNB (class in bayes.classifiers.nnb), 7

P

`partial_fit()` (bayes.classifiers.cnb.ComplementNB method), 5
`partial_fit()` (bayes.classifiers.lwnb.LocallyWeightedNB method), 7

partial_fit() (bayes.classifiers.nnb.NegationNB method), 9

partial_fit() (bayes.classifiers.snb.SelectiveNB method), 11

partial_fit() (bayes.classifiers.unb.UniversalSetNB method), 13

predict() (bayes.classifiers.cnb.ComplementNB method), 5

predict() (bayes.classifiers.lwnb.LocallyWeightedNB method), 7

predict() (bayes.classifiers.nnb.NegationNB method), 9

predict() (bayes.classifiers.snb.SelectiveNB method), 11

predict() (bayes.classifiers.unb.UniversalSetNB method), 14

predict_log_proba() (bayes.classifiers.cnb.ComplementNB method), 5

predict_log_proba() (bayes.classifiers.lwnb.LocallyWeightedNB method), 7

predict_log_proba() (bayes.classifiers.nnb.NegationNB method), 9

predict_log_proba() (bayes.classifiers.snb.SelectiveNB method), 11

predict_log_proba() (bayes.classifiers.unb.UniversalSetNB method), 14

predict_proba() (bayes.classifiers.cnb.ComplementNB method), 5

predict_proba() (bayes.classifiers.lwnb.LocallyWeightedNB method), 7

predict_proba() (bayes.classifiers.nnb.NegationNB method), 9

predict_proba() (bayes.classifiers.snb.SelectiveNB method), 12

predict_proba() (bayes.classifiers.unb.UniversalSetNB method), 14

U

UniversalSetNB (class in bayes.classifiers.unb), 12

S

safe_matmult() (bayes.classifiers.cnb.ComplementNB method), 6

safe_matmult() (bayes.classifiers.lwnb.LocallyWeightedNB method), 7

safe_matmult() (bayes.classifiers.nnb.NegationNB method), 9

safe_matmult() (bayes.classifiers.snb.SelectiveNB method), 12

safe_matmult() (bayes.classifiers.unb.UniversalSetNB method), 14

SelectiveNB (class in bayes.classifiers.snb), 10

set_params() (bayes.classifiers.cnb.ComplementNB method), 6

set_params() (bayes.classifiers.lwnb.LocallyWeightedNB method), 7

set_params() (bayes.classifiers.nnb.NegationNB method), 9