

TrafficClassification

1.0

Generated by Doxygen 1.5.5

Tue Nov 25 10:28:03 2008

Contents

1	Internet Traffic Classification Library Document	1
1.1	Introduction	1
1.2	Library Structure	1
1.3	Library Usage	1
1.4	References	1
2	Class Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	AttDesc Class Reference	9
5.2	AttDistrOnClass Class Reference	13
5.3	Attribute Class Reference	15
5.4	Classifier Class Reference	16
5.5	Dataset Class Reference	21
5.6	Distribution Class Reference	24
5.7	NaiveBayesClassifier Class Reference	25
5.8	NaiveBayesClassifierKernel Class Reference	28
5.9	NominalDistribution Class Reference	29
5.10	NormalDistribution Class Reference	30
5.11	StatisticsClassifier Class Reference	32
5.12	ValueType Union Reference	35
5.13	Xvalidator Class Reference	36

6	File Documentation	39
6.1	classifier.cpp File Reference	39
6.2	classifier.h File Reference	41
6.3	common.h File Reference	44
6.4	dataset.cpp File Reference	45
6.5	dataset.h File Reference	46
6.6	xvalidator.cpp File Reference	48
6.7	xvalidator.h File Reference	49

Chapter 1

Internet Traffic Classificationin Library Document

Author:

Kefei Lu

1.1 Introduction

1.2 Library Structure

1.3 Library Usage

1.4 References

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AttDesc	9
AttDistrOnClass	13
Attribute	15
Classifier	16
StatisticsClassifier	32
NaiveBayesClassifier	25
NaiveBayesClassifierKernel	28
Dataset	21
Distribution	24
NominalDistribution	29
NormalDistribution	30
ValueType	35
Xvalidator	36

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AttDesc (Attribute descriptor class)	9
AttDistrOnClass (A table storing distribution of the attributes conditioned on class value)	13
Attribute (Attribute instance class)	15
Classifier (Basic classifier class)	16
Dataset (The Dataset class)	21
Distribution (The distributioin's base class)	24
NaiveBayesClassifier (Naive Bayesian method)	25
NaiveBayesClassifierKernel (The class definition of the Naive Bayes Classifier with Kernel Es- timation)	28
NominalDistribution (Nominal Distribution describer)	29
NormalDistribution (Normal Distribution describer)	30
StatisticsClassifier (Classifier based on Maximum A Posteriori criteria)	32
ValueType (The value type of an attribute value)	35
Xvalidator (The cross validation structure)	36

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

classifier.cpp (Implementations of Classifier and related classes)	39
classifier.h (Classifier and related classes header file)	41
common.h (The commonly used header files in project Traffic Classification)	44
dataset.cpp (The implementations of class and methods declared in dataset.h)	45
dataset.h (Dataset and related class header file)	46
stdint.h	??
test.cpp	??
xvalidator.cpp (Implementation of cross validator)	48
xvalidator.h (Class structure description on cross validation for classifiers)	49

Chapter 5

Class Documentation

5.1 AttDesc Class Reference

[Attribute](#) descriptor class.

```
#include <dataset.h>
```

Public Member Functions

- [AttDesc](#) & **set_name** (const char *name)
- [AttDesc](#) & **set_type** (const AttType _type)
- [AttDesc](#) & **set_name_and_type** (const char *name, const AttType type)
- [AttDesc](#) & **clear** ()

Clear all the fields.

- size_t **map** (const string str) const

Map a between a stored nominal value and its index.

- size_t **map** (const char *str) const
- string **map** (const size_t index) const
- vector< string > & [possible_value_vector](#) ()

Get a reference to the possible value vector.

- const vector< string > & **possible_value_vector** () const
- [AttDesc](#) (const char *name, const AttType type)

Constructor of the type.

- [AttDesc](#) (const [AttDesc](#) &desc)

Copy constructor.

- [AttDesc](#) & **operator=** (const [AttDesc](#) &desc)

Copy assignment operator.

- [~AttDesc](#) ()

Default Destructor.

- `const char * get_name () const`
Return the name of the corresponding [Attribute](#).
- `AttType get_type () const`
Return the type of the corresponding type.

Private Attributes

- `char name [64]`
- `AttType type`
- `vector< string > * possibleValues`

5.1.1 Detailed Description

[Attribute](#) descriptor class.

This class is used to describe an attribute. It indicates the type of the attribute (`AttType`), the name of the attribute, and if the nominal type, the possible value of the attribut.

Definition at line 62 of file `dataset.h`.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `AttDesc::AttDesc (const char * name = "", const AttType type = ATT_TYPE_NONE)`

Constructor of the type.

Parameters:

A name describing the att. Take default value ""

The type of the attribute.

The `possibleValues` ptr will be assigned to a new'd address, pointing to a vector of string. It is init to an empty vector.

Definition at line 77 of file `dataset.cpp`.

5.1.2.2 `AttDesc::AttDesc (const AttDesc & desc)`

Copy constructor.

It is needed by vector's `push_back()` methods etc., because this type will need to allocate memory on constructing.

See also:

[AttDesc::operator=\(const AttDesc& desc\)](#)

Definition at line 84 of file `dataset.cpp`.

5.1.2.3 AttDesc::~~AttDesc () [inline]

Default Destructor.

Will delete the possibleValues ptr is necessary.

Definition at line 135 of file dataset.h.

5.1.3 Member Function Documentation

5.1.3.1 AttDesc & AttDesc::clear ()

Clear all the fields.

Definition at line 42 of file dataset.cpp.

5.1.3.2 size_t AttDesc::map (const string *str*) const

Map a between a stored nominal value and its index.

This function maps between a string representing its nominal value and its index in the possibleValue vector. E.g. {"a", "b", "c"}, map("c") will give 2, map(2) will give "c"

NOTE: It requires the [AttDesc](#) to be nominal.

Definition at line 51 of file dataset.cpp.

5.1.3.3 vector< string > & AttDesc::possible_value_vector ()

Get a reference to the possible value vector.

NOTE: It DOES NOT requires the [AttDesc](#) to be a nominal one to call. So don't use it on a non-nominal attribute. It makes no sense.

Definition at line 30 of file dataset.cpp.

5.1.3.4 AttDesc & AttDesc::operator= (const AttDesc & *desc*)

Copy assignment operator.

See also:

[AttDesc\(const AttDesc& desc\)](#)

Definition at line 94 of file dataset.cpp.

5.1.3.5 const char* AttDesc::get_name () const [inline]

Return the name of the corresponding [Attribute](#).

Definition at line 138 of file dataset.h.

5.1.3.6 AttType AttDesc::get_type () const `[inline]`

Return the type of the corresponding type.

Definition at line 144 of file dataset.h.

The documentation for this class was generated from the following files:

- [dataset.h](#)
- [dataset.cpp](#)

5.2 AttDistrOnClass Class Reference

A table storing distribution of the attributes conditioned on class value.

```
#include <classifier.h>
```

Public Member Functions

- void **init_table** ()
- vector< vector< [Distribution](#) * > > & **table** ()
- void **bind_classifier** (const [Classifier](#) &c)
- const [Classifier](#) & **classifier** (void)
- const double **prob** (const [ValueType](#) &value, const size_t att_i, const size_t class_j) const

Private Attributes

- const [Classifier](#) * **_classifier**
The binded [Classifier](#).
- vector< vector< [Distribution](#) * > > **_table**
[Distribution](#) of attr conditioned on class.

5.2.1 Detailed Description

A table storing distribution of the attributes conditioned on class value.

Element [r,c] corresponds to r-th class and c-th attribute, that is, the distribution information of the random variable of r-th attribute given class is c.

See also:

[NaiveBayesClassifier::_attDistrOnClass](#)

Definition at line 285 of file classifier.h.

5.2.2 Member Data Documentation

5.2.2.1 const [Classifier](#)* **AttDistrOnClass::_classifier** [private]

The binded [Classifier](#).

Definition at line 290 of file classifier.h.

5.2.2.2 vector< vector<[Distribution](#)*> > **AttDistrOnClass::_table** [private]

[Distribution](#) of attr conditioned on class.

Element [r,c] corresponds to r-th class and c-th attribute, that is, the distribution information of the random variable of r-th attribute given class is c.

Definition at line 298 of file classifier.h.

The documentation for this class was generated from the following files:

- [classifier.h](#)
- [classifier.cpp](#)

5.3 Attribute Class Reference

[Attribute](#) instance class.

```
#include <dataset.h>
```

Public Attributes

- [ValueType](#) **value**
- bool **unknown**

5.3.1 Detailed Description

[Attribute](#) instance class.

Every Instance is described by an array of attribute values. [Attribute](#) can be either a Numeric value or a Nominal value (so far). When an attribute value is unknown, the unknown flag is set to TRUE.

See also:

[Instance](#), [ValueType](#)

Definition at line 159 of file dataset.h.

The documentation for this class was generated from the following file:

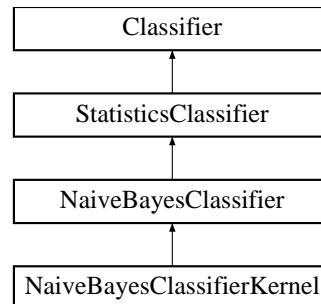
- [dataset.h](#)

5.4 Classifier Class Reference

Basic classifier class.

```
#include <classifier.h>
```

Inheritance diagram for Classifier::



Public Member Functions

- `vector< size_t > & train_set (void)`
- `const vector< size_t > & train_set (void) const`
- `vector< size_t > & test_set (void)`
- `const vector< size_t > & test_set (void) const`
- `const Dataset & dataset (void) const`
- `virtual void bind_dataset (const Dataset &dataset)`

Bind the dataset to this classifier.

- `size_t & class_index (void)`
- `const size_t & class_index (void) const`
- `bool & useAllAtt (void)`
- `const bool & useAllAtt (void) const`
- `vector< size_t > & only_these_att (void)`

Get a reference to member _onlyTheseAtt.

- `const vector< size_t > & only_these_att (void) const`
- `double & accuracy (void)`
- `vector< double > & trust (void)`
- `const vector< double > & trust (void) const`
- `ConfMatr & conf (void)`
- `const ConfMatr & conf (void) const`
- `void perf_clear (void)`

Clear the performance parameters.

- `void init_tt_set (void)`

Initialize training / testing set to the whole dataset.

- `void empty_tt_set ()`
- `const AttDesc & get_class_desc (void) const`

Get an AttDesc reference on the class attribute.

- virtual void **train** (void)=0
Train on training instances of _bindedDataset.
- void **test** (void)
Test on testing instances of _bindedDataset.
- void **show_conf** () const
Print the performance statistics.
- void **show_trust** () const
- **Classifier** (const **Dataset** &dataset, const size_t classIndex, const bool useAllAtt=1)

Private Member Functions

- virtual NominalType **classify_inst** (const **Instance** &inst, double *maxProb=NULL) const =0
Classify the instance.

Private Attributes

- vector< size_t > **_test_set**
Ratio of num of training and testing inst.
- vector< size_t > **_train_set**
- const **Dataset** * **_bindedDataset**
The dataset that this classifier is binded to This must be const.
- size_t **_classIndex**
The column of attributes that represents the class.
- vector< size_t > **_onlyTheseAtt**
Only use these attributes specified by the indecs.
- bool **_useAllAtt**
If use all the att on classification, default yes.
- double **_accuracy**
Accuracy of the whole classifier.
- **ConfMatr** **_conf**
confusion matrix
- vector< double > **_trust**
Trust for each class.

5.4.1 Detailed Description

Basic classifier class.

A simple classifier that supports cross validation on a single dataset.

Definition at line 57 of file classifier.h.

5.4.2 Member Function Documentation

5.4.2.1 `virtual NominalType Classifier::classify_inst (const Instance & inst, double * maxProb = NULL) const` `[private, pure virtual]`

Classify the instance.

Make classification on *inst*, and returns the type value. Note that *inst* may not be compatible with `_bindedDataset`, and this will not be checked by the program.

So this function is private and should only be used by other methods.

Implemented in [StatisticsClassifier](#).

5.4.2.2 `virtual void Classifier::bind_dataset (const Dataset & dataset)` `[inline, virtual]`

Bind the dataset to this classifier.

It's virtual because in the inherited Classifiers this method may be redefined so that the binding action will trigger other action, like initializing the statistics matrix, etc.

See also:

[NaiveBayesClassifier](#)

Reimplemented in [NaiveBayesClassifier](#).

Definition at line 122 of file classifier.h.

5.4.2.3 `vector<size_t>& Classifier::only_these_att (void)` `[inline]`

Get a reference to member `_onlyTheseAtt`.

When setting this member, don't forget to turn off `_useAllAtt`

See also:

[_onlyTheseAtt](#), `useAllAtt()`

Definition at line 133 of file classifier.h.

5.4.2.4 `void Classifier::perf_clear (void)` `[inline]`

Clear the performance parameters.

Definition at line 143 of file classifier.h.

5.4.2.5 void Classifier::init_tt_set (void)

Initialize training / testing set to the whole dataset.

Definition at line 33 of file classifier.cpp.

5.4.2.6 const AttDesc& Classifier::get_class_desc (void) const [inline]

Get an [AttDesc](#) reference on the class attribute.

Definition at line 154 of file classifier.h.

5.4.2.7 virtual void Classifier::train (void) [pure virtual]

Train on training instances of `_bindedDataset`.

Depends on detailed implementation.

Implemented in [StatisticsClassifier](#), and [NaiveBayesClassifier](#).

5.4.2.8 void Classifier::test (void)

Test on testing instances of `_bindedDataset`.

Definition at line 47 of file classifier.cpp.

5.4.2.9 void Classifier::show_conf () const [inline]

Print the performance statistics.

Definition at line 172 of file classifier.h.

5.4.3 Member Data Documentation**5.4.3.1 vector<size_t> Classifier::_test_set** [private]

Ratio of num of training and testing inst.

Definition at line 63 of file classifier.h.

5.4.3.2 const Dataset* Classifier::_bindedDataset [private]

The dataset that this classifier is binded to This must be const.

Definition at line 68 of file classifier.h.

5.4.3.3 size_t Classifier::_classIndex [private]

The column of attributes that represents the class.

NOTE that this must be a nominal type attribute

Definition at line 71 of file classifier.h.

5.4.3.4 `vector<size_t> Classifier::_onlyTheseAtt` [private]

Only use these attributes specified by the indecs.

Definition at line 76 of file classifier.h.

5.4.3.5 `bool Classifier::_useAllAtt` [private]

If use all the att on classification, default yes.

Definition at line 78 of file classifier.h.

5.4.3.6 `double Classifier::_accuracy` [private]

Accuracy of the whole classifier.

Definition at line 97 of file classifier.h.

5.4.3.7 `ConfMatr Classifier::_conf` [private]

confusion matrix

Definition at line 99 of file classifier.h.

5.4.3.8 `vector<double> Classifier::_trust` [private]

Trust for each class.

Definition at line 101 of file classifier.h.

The documentation for this class was generated from the following files:

- [classifier.h](#)
- [classifier.cpp](#)

5.5 Dataset Class Reference

The [Dataset](#) class.

```
#include <dataset.h>
```

Public Member Functions

- [Dataset](#) & [read_arff](#) (const char *arff_file)
Read from arff file.
- [Dataset](#) (const char *arff_file)
Init from ARFF file.
- const size_t [num_of_inst](#) () const
Get the number of instances in this dataset.
- const size_t [num_of_att](#) () const
Get the number of attributes in a instance of this dataset.
- [Instance](#) & [operator\[\]](#) (const size_t index)
Get a reference to the i-th instance.
- const [Instance](#) & [operator\[\]](#) (const size_t index) const
- [AttDesc](#) & [get_att_desc](#) (size_t index)
Return a reference to the attribute descriptor vectors (attDesc).
- const [AttDesc](#) & [get_att_desc](#) (size_t index) const
A const version of [get_att_desc\(\)](#).

Private Member Functions

- void [init](#) ()
A private init function for ctor use.

Private Attributes

- size_t [_numOfInstance](#)
- size_t [_numOfAttributes](#)
- vector< [Instance](#) > [_inst](#)
the instances in this dataset.
- vector< [AttDesc](#) > [_attDesc](#)
describe the instance structure.

5.5.1 Detailed Description

The [Dataset](#) class.

A dataset consists of: (1) an array of attribute descriptors ([AttDesc](#)), which describes each column of [Attribute](#) in the corresponding instances, for example, the type of the attribute, the possible values of the attribute (if it's a nominal one). (2) a table of instances.

It supports read in an arff file, having only nominal and numeric attributes.

This is the main class that an classification algorithm should operate on. Different algorithms should be described as different classes which can take [Dataset](#) as an argument, so that the [Dataset](#) type is acutally made reusable.

See also:

[Instance](#), [Attribute](#), [AttDesc](#)

Definition at line 192 of file dataset.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Dataset::Dataset (const char * *arff_file*)

Init from ARFF file.

Definition at line 241 of file dataset.cpp.

5.5.3 Member Function Documentation

5.5.3.1 void Dataset::init () [private]

A private init function for ctor use.

Definition at line 104 of file dataset.cpp.

5.5.3.2 Dataset & Dataset::read_arff (const char * *arff_file*)

Read from arff file.

Definition at line 114 of file dataset.cpp.

5.5.3.3 const size_t Dataset::num_of_inst () const [inline]

Get the number of instances in this dataset.

Definition at line 213 of file dataset.h.

5.5.3.4 const size_t Dataset::num_of_att () const [inline]

Get the number of attributes in a instance of this dataset.

Definition at line 216 of file dataset.h.

5.5.3.5 Instance& Dataset::operator[] (const size_t *index*) [inline]

Get a reference to the i-th instance.

By this operator and operator[] of class Instance, dataset[i][j] will return: the j-th attribute in the i-th instance in the dataset.

See also:

class [Instance](#)

Definition at line 228 of file dataset.h.

5.5.3.6 AttDesc& Dataset::get_att_desc (size_t *index*) [inline]

Return a reference to the attribute descriptor vectors (attDesc).

Definition at line 241 of file dataset.h.

5.5.3.7 const AttDesc& Dataset::get_att_desc (size_t *index*) const [inline]

A const version of [get_att_desc\(\)](#).

Definition at line 248 of file dataset.h.

5.5.4 Member Data Documentation

5.5.4.1 vector<Instance> Dataset::_inst [private]

the instances in this dataset.

Definition at line 198 of file dataset.h.

5.5.4.2 vector<AttDesc> Dataset::_attDesc [private]

describe the instance structure.

Definition at line 199 of file dataset.h.

The documentation for this class was generated from the following files:

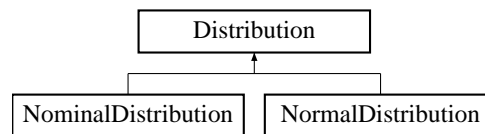
- [dataset.h](#)
- [dataset.cpp](#)

5.6 Distribution Class Reference

The distribution's base class.

```
#include <classifier.h>
```

Inheritance diagram for Distribution::



Public Member Functions

- virtual const double **prob** ([ValueType](#) value) const =0

5.6.1 Detailed Description

The distribution's base class.

There can be different kinds of distributions, each with a different way to describe. Thus the base class only provide the interface an inherited class must implement: a method that returns a probability when feed with a value.

Definition at line 231 of file classifier.h.

The documentation for this class was generated from the following file:

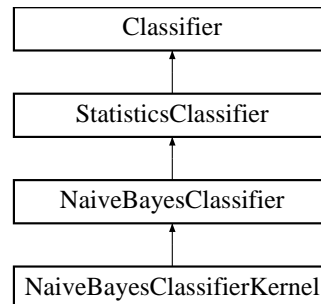
- [classifier.h](#)

5.7 NaiveBayesClassifier Class Reference

Naive Bayesian method.

```
#include <classifier.h>
```

Inheritance diagram for NaiveBayesClassifier::



Public Member Functions

- virtual void [bind_dataset](#) (const [Dataset](#) &dataset)
Bind the dataset to this classifier.
- [AttDistrOnClass](#) & [attDistrOnClass](#) (void)
- virtual void [train](#) (void)
Train the model.
- virtual const double [prob_inst_on_class](#) (const [Instance](#) &inst, const NominalType c) const
Calculate prob of an instance given a class.
- [NaiveBayesClassifier](#) (const [Dataset](#) &ds, const size_t classIndex, const bool useAllAtt=1)

Private Member Functions

- virtual double [att_prob_on_class](#) (const [ValueType](#) &value, const size_t att_i, const size_t class_j) const
Get the conditional prob of i-th att value given j-th class.
- virtual void [calc_distr_for_att_on_class](#) (size_t att_i, size_t class_j)
Calculate a [Distribution](#) for RV att_i conditioned on class_j.

Private Attributes

- [AttDistrOnClass](#) [_attDistrOnClass](#)
[Distribution](#) of attr conditioned on class.

5.7.1 Detailed Description

Naive Bayesian method.

Definition at line 320 of file classifier.h.

5.7.2 Member Function Documentation

5.7.2.1 `double NaiveBayesClassifier::att_prob_on_class (const ValueType & value, const size_t att_i, const size_t class_j) const` [private, virtual]

Get the conditional prob of i-th att value given j-th class.

This method is based on the trained model, the value of `_attDistrOnClass`, which is trained by the [train\(\)](#) method. Don't use it before the model is trained.

Definition at line 245 of file classifier.cpp.

5.7.2.2 `void NaiveBayesClassifier::calc_distr_for_att_on_class (size_t att_i, size_t class_j)` [private, virtual]

Calculate a [Distribution](#) for RV `att_i` conditioned on `class_j`.

This method is used to train the conditional probs. The obtained distribution information will be directly stored to the attribute distribution table (`attDistrOnClass()`).

When no instances belongs to this class, the `_pClass` should have been already set to 0. Set the corresponding conditional probability to invalid to indicate that when evaluating this conditional probability, 0 should be returned.

When no instances belongs to this class, for Nominal type attributes, we can assume the possible values of this attribute is equally likely to be chosen. So this zero-instance issue can be addressed the same way as the zero possibility issue stated as later. So simply set `zero_issue` flags to 1.

Handle the zero possibility issue.

$$p\{A_j|C_i\} = N(A_j, C_i) / N(C_i)$$

if $N(A_j, C_i) == 0$, means there's no such instance having A_j value and belongs to class C_i . This can be handled as:

$$p\{A_j|C_i\} = \frac{N(A_j, C_i) + 1}{N(C_i) + nPos},$$

where `nPos` is the num of possible values of this attribute.

For example, 0/3, 3/3 will become 1/5, 4/5; 0/3, 1/3, 2/3 will become 1/6, 2/6, 3/6.

Definition at line 310 of file classifier.cpp.

5.7.2.3 `void NaiveBayesClassifier::bind_dataset (const Dataset & dataset)` [virtual]

Bind the dataset to this classifier.

It's virtual because in the inherited Classifiers this method may be redefined so that the binding action will trigger other action, like initializing the statistics matrix, etc.

See also:

[NaiveBayesClassifier](#)

Reimplemented from [Classifier](#).

Definition at line 427 of file classifier.cpp.

5.7.2.4 void NaiveBayesClassifier::train (void) [virtual]

Train the model.

In here it means to estimate the `_pClass` vector.

Reimplemented from [StatisticsClassifier](#).

Definition at line 282 of file classifier.cpp.

5.7.2.5 const double NaiveBayesClassifier::prob_inst_on_class (const Instance & inst, const NominalType c) const [virtual]

Calculate prob of an instance given a class.

In [NaiveBayesClassifier](#), this is done by assuming attributes are independent to each other, and numerical attributes are normally distributed given a class label.

This may be changed by its inherited class, like `Kernel`.

Implements [StatisticsClassifier](#).

Reimplemented in [NaiveBayesClassifierKernel](#).

Definition at line 435 of file classifier.cpp.

5.7.3 Member Data Documentation

5.7.3.1 AttDistrOnClass NaiveBayesClassifier::_attDistrOnClass [private]

[Distribution](#) of attr conditioned on class.

Element `[r,c]` corresponds to `r`-th class and `c`-th attribute, that is, the distribution information of the random variable of `r`-th attribute given class is `c`.

See also:

[AttDistrOnClass](#)

Definition at line 331 of file classifier.h.

The documentation for this class was generated from the following files:

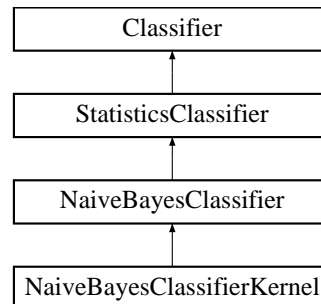
- [classifier.h](#)
- [classifier.cpp](#)

5.8 NaiveBayesClassifierKernel Class Reference

The class definition of the Naive Bayes [Classifier](#) with Kernel Estimation.

```
#include <classifier.h>
```

Inheritance diagram for NaiveBayesClassifierKernel::



Public Member Functions

- `const double prob_inst_on_class (const Instance &inst, const NominalType c) const`
Estimate the probability of a certain observation (instance) conditioned on a class label.

5.8.1 Detailed Description

The class definition of the Naive Bayes [Classifier](#) with Kernel Estimation.

The only difference to the naive Bayes classifier is the different implementation of [prob_inst_on_class\(\)](#) method, which uses kernel estimation in estimating the conditional probability.

NOT IMPLEMENTED YET.

Definition at line 393 of file classifier.h.

5.8.2 Member Function Documentation

5.8.2.1 `const double NaiveBayesClassifierKernel::prob_inst_on_class (const Instance &inst, const NominalType c) const` `[inline, virtual]`

Estimate the probability of a certain observation (instance) conditioned on a class label.

It's different from basic naive bayes in that it uses kernel estimation.

NOT IMPLEMENTED YET.

Reimplemented from [NaiveBayesClassifier](#).

Definition at line 403 of file classifier.h.

The documentation for this class was generated from the following file:

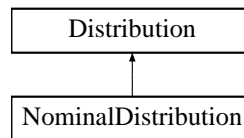
- [classifier.h](#)

5.9 NominalDistribution Class Reference

Nominal [Distribution](#) describer.

```
#include <classifier.h>
```

Inheritance diagram for NominalDistribution::



Public Member Functions

- `vector< double > & pmf ()`
- `const vector< double > & pmf () const`
- `const double prob (const ValueType value) const`

Private Attributes

- `vector< double > _pmf`

5.9.1 Detailed Description

Nominal [Distribution](#) describer.

Definition at line 267 of file classifier.h.

The documentation for this class was generated from the following file:

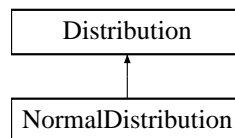
- [classifier.h](#)

5.10 NormalDistribution Class Reference

Normal [Distribution](#) describer.

```
#include <classifier.h>
```

Inheritance diagram for NormalDistribution::



Public Member Functions

- `bool & invalid ()`
- `const bool & invalid () const`
- `NumericType & mean ()`
- `const NumericType & mean () const`
- `NumericType & var ()`
- `const NumericType & var () const`
- `const double prob (const ValueType value) const`

Private Attributes

- `NumericType _mean`
- `NumericType _var`
- `bool _invalid`

By default this should be 0 to indicate a normal Normal [Distribution](#).

5.10.1 Detailed Description

Normal [Distribution](#) describer.

Specified by the mean and variance.

Definition at line 241 of file classifier.h.

5.10.2 Member Data Documentation

5.10.2.1 `bool NormalDistribution::_invalid` [private]

By default this should be 0 to indicate a normal Normal [Distribution](#).

But when the distribution is not available, i.e., no training instance can be used to train this distribution, the `_invalid` field will be set to 1, to indicate that when evaluating a probability from this distribution, 0 should be returned.

Definition at line 251 of file classifier.h.

The documentation for this class was generated from the following files:

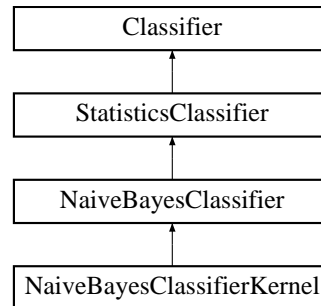
- [classifier.h](#)
- [classifier.cpp](#)

5.11 StatisticsClassifier Class Reference

[Classifier](#) based on Maximum A Posteriori criteria.

```
#include <classifier.h>
```

Inheritance diagram for StatisticsClassifier::



Public Member Functions

- `vector< double > & pClass (void)`
- `const vector< double > & pClass (void) const`
- `virtual const double prob_inst_on_class (const Instance &inst, const NominalType c) const =0`
Obtain prob of an instance given class index.
- `StatisticsClassifier (const Dataset &ds, const size_t ci, const bool useAllAtt=1)`
- `NominalType classify_inst (const Instance &inst, double *maxProb=NULL) const`
Classify the instance.
- `double a_posteriori (const NominalType c, const Instance &inst) const`
- `double likelihood (const NominalType c, const Instance &inst) const`
- `virtual void train (void)`
Train the model.

Private Member Functions

- `double est_class_prob (const size_t i) const`
Estimate the i-th class's probability.

Private Attributes

- `vector< double > _pClass`
The PMF of the class attribute random variable.

5.11.1 Detailed Description

[Classifier](#) based on Maximum A Posteriori criteria.

Definition at line 185 of file classifier.h.

5.11.2 Member Function Documentation

5.11.2.1 `double StatisticsClassifier::est_class_prob (const size_t i) const` [private]

Estimate the i-th class's probability.

ONLY use training instances. This is used for obtaining class attribute PMF.

Handling zero-instance issue (no inst. belongs to this class).

Definition at line 215 of file classifier.cpp.

5.11.2.2 `virtual const double StatisticsClassifier::prob_inst_on_class (const Instance & inst, const NominalType c) const` [pure virtual]

Obtain prob of an instance given class index.

This method has different implementation depending on which algorithm used, i.e., Naive Bayesian method with/without kernel estimation.

Implemented in [NaiveBayesClassifier](#), and [NaiveBayesClassifierKernel](#).

5.11.2.3 `NominalType StatisticsClassifier::classify_inst (const Instance & inst, double * maxProb = NULL) const` [virtual]

Classify the instance.

Make classification on inst, and returns the type value. Note that inst may not be compatible with `_bindedDataset`, and this will not be checked by the program.

So this function is private and should only be used by other methods.

Implements [Classifier](#).

Definition at line 164 of file classifier.cpp.

5.11.2.4 `void StatisticsClassifier::train (void)` [virtual]

Train the model.

In here it means to estimate the `_pClass` vector.

Implements [Classifier](#).

Reimplemented in [NaiveBayesClassifier](#).

Definition at line 259 of file classifier.cpp.

5.11.3 Member Data Documentation

5.11.3.1 `vector<double> StatisticsClassifier::_pClass` [private]

The PMF of the class attribute random variable.

Definition at line 188 of file classifier.h.

The documentation for this class was generated from the following files:

- [classifier.h](#)
- [classifier.cpp](#)

5.12 ValueType Union Reference

The value type of an attribute value.

```
#include <dataset.h>
```

Public Attributes

- NominalType **nom**
- NumericType **num**

5.12.1 Detailed Description

The value type of an attribute value.

When accessing this type, one must specify the field one wants to access, e.g. nom or num. Or the program may not interpret the type correctly.

Definition at line 41 of file dataset.h.

The documentation for this union was generated from the following file:

- [dataset.h](#)

5.13 Xvalidator Class Reference

The cross validation structure.

```
#include <xvalidator.h>
```

Public Member Functions

- [RSeed](#) & **seed** ()
- const [RSeed](#) & **seed** () const
- [Classifier](#) & **classifier** () const
- const size_t & **fold** () const
- void [set_fold](#) (const size_t f)
Re assign a fold.
- **Xvalidator** ([Classifier](#) *c, const size_t fold=3, [RSeed](#) seed=0)
- vector< vector< size_t > > & **randomIndecs** ()
- const vector< vector< size_t > > & **randomIndecs** () const
- void [init_randomIndex](#) ()
Initialize _randomIndecs.
- void [randomize](#) ()
Randomize the randomIndecs.
- void **xvalidate** ()

Private Attributes

- [RSeed](#) _seed
- [Classifier](#) * _binded_classifier
- size_t _fold
- vector< vector< size_t > > [_randomIndecs](#)
Randomized instance indecs.

5.13.1 Detailed Description

The cross validation structure.

Definition at line 16 of file xvalidator.h.

5.13.2 Member Function Documentation

5.13.2.1 void Xvalidator::set_fold (const size_t f)

Re assign a fold.

This will call [init_randomIndex\(\)](#) to re-init. the vector sizes.

See also:

[init_randomIndex\(\)](#)

Definition at line 167 of file xvalidator.cpp.

5.13.2.2 void Xvalidator::init_randomIndex ()

Initialize _randomIndecs.

Initialize the size of random indecs vectors. It must have _fold vectors, each stores a portion (nInst/_fold, except the last one.) of instance indecs, unique from the others.

See also:

[_randomIndecs](#)

Definition at line 22 of file xvalidator.cpp.

5.13.2.3 void Xvalidator::randomize (void)

Randomize the randomIndecs.

It puts (sorted) randomized indecs into the _randomIndecs vectors. It must be done before the whole cross validation process, but NOT during the process.

Definition at line 40 of file xvalidator.cpp.

5.13.3 Member Data Documentation

5.13.3.1 vector< vector<size_t> > Xvalidator::_randomIndecs [private]

Randomized instance indecs.

It must have _fold vectors, each stores a portion (nInst/_fold, except the last one.) of instance indecs, unique from the others.

Definition at line 26 of file xvalidator.h.

The documentation for this class was generated from the following files:

- [xvalidator.h](#)
- [xvalidator.cpp](#)

Chapter 6

File Documentation

6.1 classifier.cpp File Reference

Implementations of [Classifier](#) and related classes.

```
#include "classifier.h"
```

Defines

- `#define PI 3.1415926`
- `#define __CLASSIFICATION_DEBUG__`

Functions

- `void show_conf (const Classifier &c, const ConfMatr &conf)`
Print the Confusion Matrix.
- `void show_conf (const Classifier &c, const vector< vector< double > > &conf)`
This is for the average confusion matrix.
- `void show_trust (const Classifier &c, const vector< double > &trust)`
Print the trust values.
- `bool float_eq (const double v1, const double v2)`

6.1.1 Detailed Description

Implementations of [Classifier](#) and related classes.

Author:

Kefei Lu

Definition in file [classifier.cpp](#).

6.1.2 Function Documentation

6.1.2.1 `void show_conf (const Classifier & c, const vector< vector< double > > & conf)`

This is for the average confusion matrix.

Definition at line 137 of file classifier.cpp.

6.1.2.2 `void show_conf (const Classifier & c, const ConfMatr & conf)`

Print the Confusion Matrix.

Definition at line 121 of file classifier.cpp.

6.1.2.3 `void show_trust (const Classifier & c, const vector< double > & trust)`

Print the trust values.

Definition at line 152 of file classifier.cpp.

6.2 classifier.h File Reference

[Classifier](#) and related classes header file.

```
#include "common.h"
#include "dataset.h"
```

Classes

- class [Classifier](#)
Basic classifier class.
- class [StatisticsClassifier](#)
[Classifier](#) based on Maximum A Posteriori criteria.
- class [Distribution](#)
The distribution's base class.
- class [NormalDistribution](#)
Normal [Distribution](#) describer.
- class [NominalDistribution](#)
Nominal [Distribution](#) describer.
- class [AttDistrOnClass](#)
A table storing distribution of the attributes conditioned on class value.
- class [NaiveBayesClassifier](#)
Naive Bayesian method.
- class [NaiveBayesClassifierKernel](#)
The class definition of the Naive Bayes [Classifier](#) with Kernel Estimation.

Typedefs

- typedef vector< vector< size_t > > [ConfMatr](#)
Confusion Matrix.
- typedef unsigned int [RSeed](#)
Random seed type.

Functions

- void [show_conf](#) (const [Classifier](#) &c, const [ConfMatr](#) &conf)
Print the Confusion Matrix.
- void [show_conf](#) (const [Classifier](#) &c, const vector< vector< double > > &conf)

6.2.3.2 void show_conf (const Classifier & *c*, const ConfMatr & *conf*)

Print the Confusion Matrix.

Definition at line 121 of file classifier.cpp.

6.2.3.3 void show_trust (const Classifier & *c*, const vector< double > & *trust*)

Print the trust values.

Definition at line 152 of file classifier.cpp.

6.3 common.h File Reference

The commonly used header files in project Traffic Classification.

```
#include <cstdio>
#include <cstdlib>
#include <cassert>
#include <cstring>
#include <cfloat>
#include <math.h>
#include <vector>
#include <list>
#include <algorithm>
#include <string>
#include <iostream>
```

6.3.1 Detailed Description

The commonly used header files in project Traffic Classification.

Author:

Kefei Lu

See also:

[dataset.h](#) [dataset.cpp](#)

Definition in file [common.h](#).

6.4 dataset.cpp File Reference

The implementations of class and methods declared in [dataset.h](#).

```
#include "dataset.h"
```

Defines

- `#define MAX_LINE_CHAR 20000`

6.4.1 Detailed Description

The implementations of class and methods declared in [dataset.h](#).

Author:

Kefei Lu

See also:

[dataset.h](#)

Definition in file [dataset.cpp](#).

6.5 dataset.h File Reference

[Dataset](#) and related class header file.

```
#include "common.h"
```

Classes

- union [ValueType](#)
The value type of an attribute value.
- class [AttDesc](#)
Attribute descriptor class.
- class [Attribute](#)
Attribute instance class.
- class [Dataset](#)
The [Dataset](#) class.

Typedefs

- typedef uint64_t **NominalType**
- typedef double **NumericType**
- typedef enum [_AttType](#) **AttType**
- typedef vector< [Attribute](#) > [Instance](#)
Instance type.

Enumerations

- enum [_AttType](#) { [ATT_TYPE_NONE](#) = 0, [ATT_TYPE_NOMINAL](#), [ATT_TYPE_NUMERIC](#) }
Attribute Type.

6.5.1 Detailed Description

[Dataset](#) and related class header file.

Author:

Kefei Lu This file contains classes that consists of the [Dataset](#) class.

See also:

[dataset.cpp](#)

Definition in file [dataset.h](#).

6.5.2 Typedef Documentation

6.5.2.1 typedef vector<Attribute> Instance

Instance type.

Instance is described by an array of Attributes.

Definition at line 170 of file dataset.h.

6.5.3 Enumeration Type Documentation

6.5.3.1 enum _AttType

[Attribute](#) Type.

Enumerator:

ATT_TYPE_NONE A padding one.

ATT_TYPE_NOMINAL Nominal type.

ATT_TYPE_NUMERIC Numeric type.

Definition at line 49 of file dataset.h.

6.6 xvalidator.cpp File Reference

Implementation of cross validator.

```
#include "common.h"
#include "xvalidator.h"
```

Functions

- static double **double_sum** (const double v1, const double v2)
- static size_t **size_t_sum** (const size_t v1, const size_t v2)
- static vector< double > & **vector_sum** (vector< double > &v1, const vector< size_t > &v2)

6.6.1 Detailed Description

Implementation of cross validator.

Author:

Kefei Lu

Definition in file [xvalidator.cpp](#).

6.7 xvalidator.h File Reference

Class structure description on cross validation for classifiers.

```
#include "common.h"
#include "dataset.h"
#include "classifier.h"
```

Classes

- class [Xvalidator](#)

The cross validation structure.

6.7.1 Detailed Description

Class structure description on cross validation for classifiers.

Author:

Kefei Lu

Definition in file [xvalidator.h](#).

Index

- ~AttDesc
 - AttDesc, [10](#)
- _AttType
 - dataset.h, [47](#)
- _accuracy
 - Classifier, [20](#)
- _attDesc
 - Dataset, [23](#)
- _attDistrOnClass
 - NaiveBayesClassifier, [27](#)
- _bindedDataset
 - Classifier, [19](#)
- _classIndex
 - Classifier, [19](#)
- _classifier
 - AttDistrOnClass, [13](#)
- _conf
 - Classifier, [20](#)
- _inst
 - Dataset, [23](#)
- _invalid
 - NormalDistribution, [30](#)
- _onlyTheseAtt
 - Classifier, [19](#)
- _pClass
 - StatisticsClassifier, [34](#)
- _randomIndecs
 - Xvalidator, [37](#)
- _table
 - AttDistrOnClass, [13](#)
- _test_set
 - Classifier, [19](#)
- _trust
 - Classifier, [20](#)
- _useAllAtt
 - Classifier, [20](#)
- ATT_TYPE_NOMINAL
 - dataset.h, [47](#)
- ATT_TYPE_NONE
 - dataset.h, [47](#)
- ATT_TYPE_NUMERIC
 - dataset.h, [47](#)
- att_prob_on_class
 - NaiveBayesClassifier, [26](#)
- AttDesc, [9](#)
 - ~AttDesc, [10](#)
 - AttDesc, [10](#)
 - clear, [11](#)
 - get_name, [11](#)
 - get_type, [11](#)
 - map, [11](#)
 - operator=, [11](#)
 - possible_value_vector, [11](#)
- AttDistrOnClass, [13](#)
 - _classifier, [13](#)
 - _table, [13](#)
- Attribute, [15](#)
- bind_dataset
 - Classifier, [18](#)
 - NaiveBayesClassifier, [26](#)
- calc_distr_for_att_on_class
 - NaiveBayesClassifier, [26](#)
- Classifier, [16](#)
 - _accuracy, [20](#)
 - _bindedDataset, [19](#)
 - _classIndex, [19](#)
 - _conf, [20](#)
 - _onlyTheseAtt, [19](#)
 - _test_set, [19](#)
 - _trust, [20](#)
 - _useAllAtt, [20](#)
 - bind_dataset, [18](#)
 - classify_inst, [18](#)
 - get_class_desc, [19](#)
 - init_tt_set, [18](#)
 - only_these_att, [18](#)
 - perf_clear, [18](#)
 - show_conf, [19](#)
 - test, [19](#)
 - train, [19](#)
- classifier.cpp, [39](#)
 - show_conf, [40](#)
 - show_trust, [40](#)
- classifier.h, [41](#)
 - ConfMatr, [42](#)
 - RSeed, [42](#)
 - show_conf, [42](#)

- show_trust, 43
- classify_inst
 - Classifier, 18
 - StatisticsClassifier, 33
- clear
 - AttDesc, 11
- common.h, 44
- ConfMatr
 - classifier.h, 42
- Dataset, 21
 - _attDesc, 23
 - _inst, 23
 - Dataset, 22
 - get_att_desc, 23
 - init, 22
 - num_of_att, 22
 - num_of_inst, 22
 - read_arff, 22
- dataset.cpp, 45
- dataset.h, 46
 - _AttType, 47
 - ATT_TYPE_NOMINAL, 47
 - ATT_TYPE_NONE, 47
 - ATT_TYPE_NUMERIC, 47
 - Instance, 47
- Distribution, 24
- est_class_prob
 - StatisticsClassifier, 33
- get_att_desc
 - Dataset, 23
- get_class_desc
 - Classifier, 19
- get_name
 - AttDesc, 11
- get_type
 - AttDesc, 11
- init
 - Dataset, 22
- init_randomIndex
 - Xvalidator, 37
- init_tt_set
 - Classifier, 18
- Instance
 - dataset.h, 47
- map
 - AttDesc, 11
- NaiveBayesClassifier, 25
 - _attDistrOnClass, 27
 - att_prob_on_class, 26
 - bind_dataset, 26
 - calc_distr_for_att_on_class, 26
 - prob_inst_on_class, 27
 - train, 27
- NaiveBayesClassifierKernel, 28
 - prob_inst_on_class, 28
- NominalDistribution, 29
- NormalDistribution, 30
 - _invalid, 30
- num_of_att
 - Dataset, 22
- num_of_inst
 - Dataset, 22
- only_these_att
 - Classifier, 18
- operator=
 - AttDesc, 11
- perf_clear
 - Classifier, 18
- possible_value_vector
 - AttDesc, 11
- prob_inst_on_class
 - NaiveBayesClassifier, 27
 - NaiveBayesClassifierKernel, 28
 - StatisticsClassifier, 33
- randomize
 - Xvalidator, 37
- read_arff
 - Dataset, 22
- RSeed
 - classifier.h, 42
- set_fold
 - Xvalidator, 36
- show_conf
 - Classifier, 19
 - classifier.cpp, 40
 - classifier.h, 42
- show_trust
 - classifier.cpp, 40
 - classifier.h, 43
- StatisticsClassifier, 32
 - _pClass, 34
 - classify_inst, 33
 - est_class_prob, 33
 - prob_inst_on_class, 33
 - train, 33
- test
 - Classifier, 19
- train
 - Classifier, 19

NaiveBayesClassifier, [27](#)
StatisticsClassifier, [33](#)

ValueType, [35](#)

Xvalidator, [36](#)
 _randomIndecs, [37](#)
 init_randomIndex, [37](#)
 randomize, [37](#)
 set_fold, [36](#)
xvalidator.cpp, [48](#)
xvalidator.h, [49](#)