

# RFlib Reference Manual

## 1.0

Generated by Doxygen 1.5.4

Tue Mar 2 17:27:59 2010



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>RFlib Hierarchical Index</b>   | <b>1</b>  |
| 1.1      | RFlib Class Hierarchy . . . . .   | 1         |
| <b>2</b> | <b>RFlib Class Index</b>  | <b>3</b>  |
| 2.1      | RFlib Class List . . . . .  | 3         |
| <b>3</b> | <b>RFlib File Index</b>   | <b>5</b>  |
| 3.1      | RFlib File List . . . . .   | 5         |
| <b>4</b> | <b>RFlib Class Documentation</b>  | <b>7</b>  |
| 4.1      | Color Class Reference . . . . .   | 7         |
| 4.2      | Histogram Class Reference . . . . .   | 11        |
| 4.3      | HistogramAssignmentError Class Reference . . . . .                                    | 18        |
| 4.4      | HistogramOperationError Class Reference . . . . .                                     | 19        |
| 4.5      | IClassifier Class Reference . . . . .   | 20        |
| 4.6      | IFeatureVector Class Reference . . . . .  | 24        |
| 4.7      | LeafNode Class Reference . . . . .  | 25        |
| 4.8      | Patch Class Reference . . . . .   | 28        |
| 4.9      | RandomizedForest Class Reference . . . . .  | 32        |
| 4.10     | RandomizedTest Class Reference . . . . .  | 37        |
| 4.11     | RandomizedTree Class Reference . . . . .  | 40        |
| <b>5</b> | <b>RFlib File Documentation</b>   | <b>47</b> |
| 5.1      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Color.h File Reference . . . . .            | 47        |
| 5.2      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Histogram.h File Reference . . . . .        | 48        |
| 5.3      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IClassifier.h File Reference . . . . .      | 49        |
| 5.4      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IFeatureVector.h File Reference . . . . .   | 50        |
| 5.5      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/LeafNode.h File Reference . . . . .         | 51        |
| 5.6      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h File Reference . . . . .            | 52        |
| 5.7      | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h File Reference . . . . . | 53        |

|      |   |    |
|------|---|----|
| 5.8  | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTest.h File Reference . . . . .     | 54 |
| 5.9  | C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTree.h File Reference . . . . .     | 55 |
| 5.10 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Color.cpp File Reference . . . . .            | 56 |
| 5.11 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Histogram.cpp File Reference . . . . .        | 57 |
| 5.12 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IClassifier.cpp File Reference . . . . .      | 58 |
| 5.13 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IFeatureVector.cpp File Reference . . . . .   | 59 |
| 5.14 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/LeafNode.cpp File Reference . . . . .         | 60 |
| 5.15 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Patch.cpp File Reference . . . . .            | 61 |
| 5.16 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedForest.cpp File Reference . . . . . | 62 |
| 5.17 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTest.cpp File Reference . . . . .   | 63 |
| 5.18 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTree.cpp File Reference . . . . .   | 64 |
| 5.19 | C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run_patch.cpp File Reference . . . . .        | 65 |

# Chapter 1

## RFlib Hierarchical Index

### 1.1 RFlib Class Hierarchy

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

|                                    |    |
|------------------------------------|----|
| Color . . . . .                    | 7  |
| Histogram . . . . .                | 11 |
| HistogramAssignmentError . . . . . | 18 |
| HistogramOperationError . . . . .  | 19 |
| IClassifier . . . . .              | 20 |
| RandomizedForest . . . . .         | 32 |
| IFeatureVector . . . . .           | 24 |
| Patch . . . . .                    | 28 |
| RandomizedTest . . . . .           | 37 |
| RandomizedTree . . . . .           | 40 |
| LeafNode . . . . .                 | 25 |



## Chapter 2

# RFlib Class Index

### 2.1 RFlib Class List

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

|                                 |    |
|---------------------------------|----|
| <b>Color</b>                    | 7  |
| <b>Histogram</b>                | 11 |
| <b>HistogramAssignmentError</b> | 18 |
| <b>HistogramOperationError</b>  | 19 |
| <b>IClassifier</b>              | 20 |
| <b>IFeatureVector</b>           | 24 |
| <b>LeafNode</b>                 | 25 |
| <b>Patch</b>                    | 28 |
| <b>RandomizedForest</b>         | 32 |
| <b>RandomizedTest</b>           | 37 |
| <b>RandomizedTree</b>           | 40 |





## Chapter 3

# RFlib File Index

### 3.1 RFlib File List

Here is a list of all files with brief descriptions:

|  |    |
|--|----|
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Color.h . . . . .              | 47 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Histogram.h . . . . .          | 48 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IClassifier.h . . . . .        | 49 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IFeatureVector.h . . . . .     | 50 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/LeafNode.h . . . . .           | 51 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h . . . . .              | 52 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h . . . . .   | 53 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTest.h . . . . .     | 54 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTree.h . . . . .     | 55 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Color.cpp . . . . .            | 56 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Histogram.cpp . . . . .        | 57 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IClassifier.cpp . . . . .      | 58 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IFeatureVector.cpp . . . . .   | 59 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/LeafNode.cpp . . . . .         | 60 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Patch.cpp . . . . .            | 61 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedForest.cpp . . . . . | 62 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTest.cpp . . . . .   | 63 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTree.cpp . . . . .   | 64 |
| C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run_patch.cpp . . . . .        | 65 |



## Chapter 4

# RFlib Class Documentation

### 4.1 Color Class Reference

```
#include <Color.h>
```

#### Public Member Functions

- **Color** ()  
*Default constructor : white.*
- **Color** (int r, int g, int b, double alpha=1.)  
*Constructor with int R,G,B.*
- **Color** (double d, double alpha=1.)
- **Color** (std::string name, double alpha=1.)  
*Constructor with a string as the name of the color.*
- **Color** (const **Color** &other)  
*Copy Constructor.*
- **~Color** ()  
*Destructor.*
- bool **operator==** (const **Color** &other) const  
*Overload of the equal operator==.*
- bool **operator!=** (const **Color** &other) const  
*Overload of the equal operator!=.*
- const unsigned char **getR** () const  
*Accessor to the red component.*
- const unsigned char **getG** () const  
*Accessor to the green component.*

- const unsigned char **getB** () const  
*Accessor to the blue component.*
- const double **getAlpha** () const  
*get the opacity value alpha*
- **Color & operator=** (const **Color** &other)  
*Set the color : overload the assignation operator.*
- **Color & rainbow** (double x)
- **Color & rainbow\_cube** (double x)
- **Color & random** ()  
*Make a random rainbow color.*
- void **setAlpha** (double alpha)  
*Set the alpha channel.*
- std::string **toString** () const
- bool **isDefault** () const

## Static Public Member Functions

- static void **rgb2hsv** (int r, int g, int b, int &h, double &s, double &v)
- static void **hsv2rgb** (int h, double s, double v, int &r, int &g, int &b)

## Private Member Functions

- void **setR** (int r)
- void **setG** (int g)
- void **setB** (int b)

## Private Attributes

- unsigned char **m\_R**
- unsigned char **m\_G**
- unsigned char **m\_B**
- double **m\_alpha**

### 4.1.1 Constructor & Destructor Documentation

#### 4.1.1.1 Color::Color ()

Default constructor : white.

#### 4.1.1.2 Color::Color (int r, int g, int b, double alpha = 1 . )

Constructor with int R,G,B.

**4.1.1.3 Color::Color (double *d*, double *alpha* = 1 .)**

Constructor with a double : give a rainbow color negative is black, greater than one is white, in between is the rainbow.

**4.1.1.4 Color::Color (std::string *name*, double *alpha* = 1 .)**

Constructor with a string as the name of the color.

**4.1.1.5 Color::Color (const Color & *other*)**

Copy Constructor.

**4.1.1.6 Color::~~Color ()**

Destructor.

**4.1.2 Member Function Documentation****4.1.2.1 bool Color::operator== (const Color & *other*) const**

Overload of the equal operator==.

**4.1.2.2 bool Color::operator!= (const Color & *other*) const**

Overload of the equal operator!=.

**4.1.2.3 const unsigned char Color::getR () const**

Accessor to the red component.

**4.1.2.4 const unsigned char Color::getG () const**

Accessor to the green component.

**4.1.2.5 const unsigned char Color::getB () const**

Accessor to the blue component.

**4.1.2.6 const double Color::getAlpha () const**

get the opacity value alpha

**4.1.2.7 Color & Color::operator= (const Color & *other*)**

Set the color : overload the assignation operator.

#### 4.1.2.8 Color & Color::rainbow (double *x*)

Set the color as a rainbow color, defined by a float between 0 and 1 (based on hsv, from 0 to 270°) Lower than 0 is black, greater than one is white

#### 4.1.2.9 Color & Color::rainbow\_cube (double *x*)

Set the color as a rainbow color, defined by a float between 0 and 1 based on moving along the edges of the rgb cube Lower than 0 is black, greater than one is white

#### 4.1.2.10 Color & Color::random ()

Make a random rainbow color.

#### 4.1.2.11 void Color::setAlpha (double *alpha*)

Set the alpha channel.

#### 4.1.2.12 std::string Color::toString () const

#### 4.1.2.13 bool Color::isDefault () const

#### 4.1.2.14 void Color::rgb2hsv (int *r*, int *g*, int *b*, int & *h*, double & *s*, double & *v*) [static]

#### 4.1.2.15 void Color::hsv2rgb (int *h*, double *s*, double *v*, int & *r*, int & *g*, int & *b*) [static]

#### 4.1.2.16 void Color::setR (int *r*) [private]

#### 4.1.2.17 void Color::setG (int *g*) [private]

#### 4.1.2.18 void Color::setB (int *b*) [private]

### 4.1.3 Member Data Documentation

#### 4.1.3.1 unsigned char Color::m\_R [private]

#### 4.1.3.2 unsigned char Color::m\_G [private]

#### 4.1.3.3 unsigned char Color::m\_B [private]

#### 4.1.3.4 double Color::m\_alpha [private]

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

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Color.h
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Color.cpp

## 4.2 Histogram Class Reference

```
#include <Histogram.h>
```

### Public Member Functions

- **Histogram** (unsigned int nbClasses=2)  
*Constructor given a number of classes.*
- **Histogram** (unsigned int nbClasses, double \*data)  
*constructor given the number of classes and the data itself*
- **Histogram** (const **Histogram** &other)  
*Copy constructor.*
- virtual ~**Histogram** ()  
*Destructor.*
- **Histogram** & **operator=** (const **Histogram** &other)  
*Assignement operator.*
- **Histogram** & **zeros** ()  
*Fill the histogram with a given value.*
- **Histogram** & **ones** ()
- **Histogram** & **fill** (double val)
- **Histogram** & **basis** (int i)
- double **sum** () const  
*Get the sum of the value of the histogram.*
- double **maxValue** () const  
*Compute the max of the histogram.*
- unsigned int **argMax** () const  
*Compute the argmax of the histogram.*
- double **minValue** () const  
*compute the min value*
- unsigned int **argMin** () const  
*compute the index where the histogram is minimal*
- unsigned int **getNbClasses** () const  
*get the number of classes*
- const double & **operator[ ]** (int i) const  
*get a specific value of the histogram*
- double & **operator[ ]** (int i)

*the same, but enable to modify the value*

- double **norm** () const  
*Get the L2 norm.*
- double **distanceL1** (const **Histogram** &other) const  
*distance between histograms*
- double **distanceL2** (const **Histogram** &other) const
- double **distanceLP** (const **Histogram** &other, int p) const
- double **kl** (const **Histogram** &other) const
- double **battacharyya** (const **Histogram** &other) const
- double **hellinger** (const **Histogram** &other) const
- **Histogram & normalize** ()  
*Normalize the histogram.*
- **Histogram & normalize2** ()  
*Normalize the histogram with L2 norm.*
- **Histogram & operator+=** (const **Histogram** &other)  
*Add two histograms.*
- **Histogram operator+** (const **Histogram** other)
- **Histogram & operator \*=** (const **Histogram** &other)  
*Mutlification element-wise of two histograms.*
- **Histogram operator \*** (const **Histogram** other)
- **Histogram & operator/=** (const **Histogram** &other)  
*Division element-wise of two histograms (keep 0 if the divider is 0).*
- **Histogram operator/** (const **Histogram** other)
- **Histogram & scale** (double s)  
*Scale an histogram.*
- **Histogram & linearCombination** (const std::vector< **Histogram** > &vectors, const std::vector< double > &coeffs)  
*set the histogram as a linear combination of histograms*
- double **dot** (const **Histogram** &other) const  
*Compute the dot product of two vectors.*
- double **getWeight** () const  
*Accessor to the weight.*
- void **setWeight** (double w)  
*Set the value of the weight.*
- void **resetWeight** ()  
*reset the value of the weight to one*



- **Histogram & resize** (unsigned int k)  
*Resize the histogram, by truncating or zero padding.*
- unsigned int **sample** () const  
*interpret the values of the histogram as a pdf, and sample a index according to it*
- **Histogram & egreedy** (double epsilon, unsigned int k)  
*set the histogram as a e-greedy distribution on a peak value k*

## Protected Attributes

- double \* **m\_data**  
*stores the histogram itself*
- unsigned int **m\_nb\_classes**  
*the given number of classes*
- double **m\_weight**  
*Weight of the histogram : useful while doing sums of normalized histograms.*

## Friends

- std::ostream & **operator<<** (std::ostream &out, const **Histogram** &hist)  
*dump the histogram into a ofstream, to be save or plotted*
- std::istream & **operator>>** (std::istream &in, **Histogram** &hist)

### 4.2.1 Constructor & Destructor Documentation

#### 4.2.1.1 **Histogram::Histogram** (unsigned int nbClasses = 2) [explicit]

Constructor given a number of classes.

#### 4.2.1.2 **Histogram::Histogram** (unsigned int nbClasses, double \* data)

constructor given the number of classes and the data itself

#### 4.2.1.3 **Histogram::Histogram** (const **Histogram** & other)

Copy constructor.

#### 4.2.1.4 **Histogram::~Histogram** () [virtual]

Destructor.

## 4.2.2 Member Function Documentation

### 4.2.2.1 Histogram & Histogram::operator= (const Histogram & *other*)

Assignment operator.

### 4.2.2.2 Histogram & Histogram::zeros ()

Fill the histogram with a given value.

### 4.2.2.3 Histogram & Histogram::ones ()

### 4.2.2.4 Histogram & Histogram::fill (double *val*)

### 4.2.2.5 Histogram & Histogram::basis (int *i*)

### 4.2.2.6 double Histogram::sum () const

Get the sum of the value of the histogram.

### 4.2.2.7 double Histogram::maxValue () const

Compute the max of the histogram.

### 4.2.2.8 unsigned int Histogram::argMax () const

Compute the argmax of the histogram.

### 4.2.2.9 double Histogram::minValue () const

compute the min value

### 4.2.2.10 unsigned int Histogram::argMin () const

compute the index where the histogram is minimal

### 4.2.2.11 unsigned int Histogram::getNbClasses () const

get the number of classes

### 4.2.2.12 const double & Histogram::operator[] (int *i*) const

get a specific value of the histogram

### 4.2.2.13 double & Histogram::operator[] (int *i*)

the same, but enable to modify the value

**4.2.2.14 double Histogram::norm () const**

Get the L2 norm.

**4.2.2.15 double Histogram::distanceL1 (const Histogram & *other*) const**

distance between histograms

**4.2.2.16 double Histogram::distanceL2 (const Histogram & *other*) const****4.2.2.17 double Histogram::distanceLP (const Histogram & *other*, int *p*) const****4.2.2.18 double Histogram::kl (const Histogram & *other*) const****4.2.2.19 double Histogram::battacharyya (const Histogram & *other*) const****4.2.2.20 double Histogram::hellinger (const Histogram & *other*) const****4.2.2.21 Histogram & Histogram::normalize ()**

Normalize the histogram.

**4.2.2.22 Histogram & Histogram::normalize2 ()**

Normalize the histogram with L2 norm.

**4.2.2.23 Histogram & Histogram::operator+= (const Histogram & *other*)**

Add two histograms.

**4.2.2.24 Histogram Histogram::operator+ (const Histogram *other*)****4.2.2.25 Histogram & Histogram::operator \*= (const Histogram & *other*)**

Mutliplication element-wise of two histograms.

**4.2.2.26 Histogram Histogram::operator \* (const Histogram *other*)****4.2.2.27 Histogram & Histogram::operator/= (const Histogram & *other*)**

Division element-wise of two histograms (keep 0 if the divider is 0).

**4.2.2.28 Histogram Histogram::operator/ (const Histogram *other*)****4.2.2.29 Histogram & Histogram::scale (double *s*)**

Scale an histogram.

#### 4.2.2.30 **Histogram & Histogram::linearCombination** (const std::vector< Histogram > & *vectors*, const std::vector< double > & *coeffs*)

set the histogram as a linear combination of histograms

#### 4.2.2.31 **double Histogram::dot** (const Histogram & *other*) const

Compute the dot product of two vectors.

#### 4.2.2.32 **double Histogram::getWeight** () const

Accessor to the weight.

#### 4.2.2.33 **void Histogram::setWeight** (double *w*)

Set the value of the weight.

#### 4.2.2.34 **void Histogram::resetWeight** ()

reset the value of the weight to one

#### 4.2.2.35 **Histogram & Histogram::resize** (unsigned int *k*)

Resize the histogram, by truncating or zero padding.

#### 4.2.2.36 **unsigned int Histogram::sample** () const

interpret the values of the histogram as a pdf, and sample a index according to it

#### 4.2.2.37 **Histogram & Histogram::egreedy** (double *epsilon*, unsigned int *k*)

set the histogram as a e-greedy distribution on a peak value k

### 4.2.3 Friends And Related Function Documentation

#### 4.2.3.1 **std::ostream& operator<<** (std::ostream & *out*, const Histogram & *hist*) [friend]

dump the histogram into a ofstream, to be save or plotted

#### 4.2.3.2 **std::istream& operator>>** (std::istream & *in*, Histogram & *hist*) [friend]

### 4.2.4 Member Data Documentation

#### 4.2.4.1 **double\* Histogram::m\_data** [protected]

stores the histogram itself

**4.2.4.2 unsigned int Histogram::m\_nb\_classes** [protected]

the given number of classes

**4.2.4.3 double Histogram::m\_weight** [protected]

Weight of the histogram : useful while doing sums of normalized histograms.

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

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**Histogram.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**Histogram.cpp**

## 4.3 HistogramAssignmentError Class Reference

```
#include <Histogram.h>
```

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

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**Histogram.h**

## 4.4 HistogramOperationError Class Reference

```
#include <Histogram.h>
```

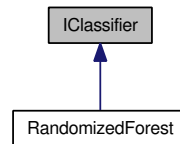
The documentation for this class was generated from the following file:

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**Histogram.h**

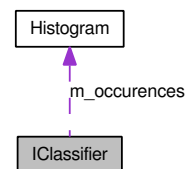
## 4.5 IClassifier Class Reference

```
#include <IClassifier.h>
```

Inheritance diagram for IClassifier:



Collaboration diagram for IClassifier:



### Public Member Functions

- **IClassifier** (unsigned int nbLabels=2, bool occurrence\_normalization=false)  
*Constructor.*
- **IClassifier** (const **IClassifier** &other)  
*Copy constructor.*
- virtual ~**IClassifier** ()  
*Destructor.*
- virtual void **train** (const **IFeatureVector** &vector, unsigned int label)=0  
*virtual method to train a classifier from a patch*
- virtual **Histogram** **estimatePosterior** (const **IFeatureVector** &vector) const =0  
*Estimate posterior probability of a patch.*
- virtual void **load** (const std::string &filename)=0  
*virtual method to load a classifier from a file*
- virtual void **save** (const std::string &filename)=0  
*virtual method to save a classifier in a file*
- virtual **IClassifier** \* **clone** () const =0  
*Clone the classifier.*
- virtual void **reset** ()=0  
*Clean the classifier, so that it forgets whatever it has learnt before.*



- unsigned int **classify** (const **IFeatureVector** &vector) const
- unsigned int **getNbLabels** () const  
*Get the number of label the classifier handles.*
- void **setOccurenceNormalization** (bool on)  
*set the occurence normalization*

## Protected Attributes

- unsigned int **m\_nb\_labels**  
*Number of considered classes.*
- **Histogram m\_occurences**  
*Keep track of the number of encountered label during training. Can be used to balance the training data.*
- bool **m\_occurence\_normalization**  
*Use occurence normalization while testing ?*

## 4.5.1 Constructor & Destructor Documentation

### 4.5.1.1 IClassifier::IClassifier (unsigned int nbLabels = 2, bool occurence\_normalization = false)

Constructor.

### 4.5.1.2 IClassifier::IClassifier (const IClassifier & other)

Copy constructor.

### 4.5.1.3 IClassifier::~~IClassifier () [virtual]

Destructor.

## 4.5.2 Member Function Documentation

### 4.5.2.1 virtual void IClassifier::train (const IFeatureVector & vector, unsigned int label) [pure virtual]

virtual method to train a classifier from a patch

Implemented in **RandomizedForest** (p. 34).

### 4.5.2.2 virtual Histogram IClassifier::estimatePosterior (const IFeatureVector & vector) const [pure virtual]

Estimate posterior probability of a patch.

Implemented in **RandomizedForest** (p. 34).

**4.5.2.3 virtual void IClassifier::load (const std::string & *filename*)** [pure virtual]

virtual method to load a classifier from a file

Implemented in **RandomizedForest** (p. 34).

**4.5.2.4 virtual void IClassifier::save (const std::string & *filename*)** [pure virtual]

virtual method to save a classifier in a file

Implemented in **RandomizedForest** (p. 35).

**4.5.2.5 virtual IClassifier\* IClassifier::clone () const** [pure virtual]

Clone the classifier.

Implemented in **RandomizedForest** (p. 35).

**4.5.2.6 virtual void IClassifier::reset ()** [pure virtual]

Clean the classifier, so that it forgets whatever it has learnt before.

Implemented in **RandomizedForest** (p. 35).

**4.5.2.7 unsigned int IClassifier::classify (const IFeatureVector & *vector*) const****4.5.2.8 unsigned int IClassifier::getNbLabels () const**

Get the number of label the classifier handles.

**4.5.2.9 void IClassifier::setOccurenceNormalization (bool *on*)**

set the occurence normalization

**4.5.3 Member Data Documentation****4.5.3.1 unsigned int IClassifier::m\_nb\_labels** [protected]

Number of considered classes.

**4.5.3.2 Histogram IClassifier::m\_occurences** [protected]

Keep track of the number of encountered label during training. Can be used to balance the training data.

**4.5.3.3 bool IClassifier::m\_occurence\_normalization** [protected]

Use occurence normalization while testing ?

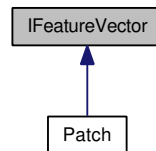
The documentation for this class was generated from the following files:

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**IClassifier.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**IClassifier.cpp**

## 4.6 IFeatureVector Class Reference

```
#include <IFeatureVector.h>
```

Inheritance diagram for IFeatureVector:



### Public Member Functions

- **IFeatureVector** ()
- **IFeatureVector** (const **IFeatureVector** &other)
- virtual **~IFeatureVector** ()
- virtual **IFeatureVector** \* **clone** () const =0
- virtual double **operator**[ ] (unsigned int k) const =0
- virtual unsigned int **size** () const =0

### 4.6.1 Constructor & Destructor Documentation

**4.6.1.1** **IFeatureVector::IFeatureVector** ()

**4.6.1.2** **IFeatureVector::IFeatureVector** (const **IFeatureVector** & *other*)

**4.6.1.3** **IFeatureVector::~~IFeatureVector** () [virtual]

### 4.6.2 Member Function Documentation

**4.6.2.1** virtual **IFeatureVector**\* **IFeatureVector::clone** () const [pure virtual]

Implemented in **Patch** (p. 30).

**4.6.2.2** virtual double **IFeatureVector::operator**[ ] (unsigned int *k*) const [pure virtual]

Implemented in **Patch** (p. 30).

**4.6.2.3** virtual unsigned int **IFeatureVector::size** () const [pure virtual]

Implemented in **Patch** (p. 30).

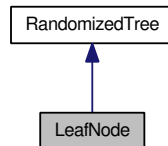
The documentation for this class was generated from the following files:

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**IFeatureVector.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**IFeatureVector.cpp**

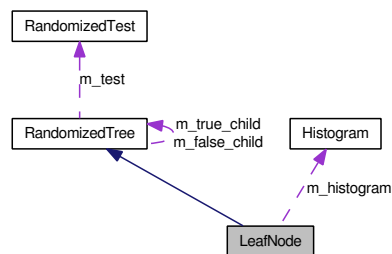
## 4.7 LeafNode Class Reference

```
#include <LeafNode.h>
```

Inheritance diagram for LeafNode:



Collaboration diagram for LeafNode:



### Public Member Functions

- **LeafNode** (unsigned int nbLabels)  
*Constructor.*
- **LeafNode** (const **LeafNode** &other)  
*Copy Constructor.*
- void **copy** (**RandomizedTree** \*other)  
*Copy a leaf node into another one (cast).*
- **~LeafNode** ()  
*Destructor.*
- bool **isLeaf** () const  
*A leaf is... a leaf of course, what do you think O\_o ?*
- const **Histogram** & **getHistogram** () const  
*Get the histogram.*
- const double & **operator** [ ] (unsigned int i) const  
*Accessor directly to the ith bin of the histogram.*
- double & **operator** [ ] (unsigned int i)  
*the same version, but that allows to modify the ith bin*

## Private Member Functions

- void **train** (const **IFeatureVector** &vector, int label, int d)

## Private Attributes

- **Histogram** \* **m\_histogram**

## 4.7.1 Constructor & Destructor Documentation

### 4.7.1.1 LeafNode::LeafNode (unsigned int *nbLabels*)

Constructor.

### 4.7.1.2 LeafNode::LeafNode (const LeafNode & *other*)

Copy Constructor.

### 4.7.1.3 LeafNode::~~LeafNode ()

Destructor.

## 4.7.2 Member Function Documentation

### 4.7.2.1 void LeafNode::copy (RandomizedTree \* *other*) [virtual]

Copy a leaf node into another one (cast).

Reimplemented from **RandomizedTree** (p. 42).

### 4.7.2.2 bool LeafNode::isLeaf () const [virtual]

A leaf is... a leaf of course, what do you think O\_o ?

Reimplemented from **RandomizedTree** (p. 43).

### 4.7.2.3 const Histogram & LeafNode::getHistogram () const

Get the histogram.

### 4.7.2.4 const double & LeafNode::operator[] (unsigned int *i*) const

Accessor directly to the *i*th bin of the histogram.

### 4.7.2.5 double & LeafNode::operator[] (unsigned int *i*)

the same version, but that allows to modify the *i*th bin

**4.7.2.6** `void LeafNode::train (const IFeatureVector & vector, int label, int d)` [private, virtual]

Reimplemented from **RandomizedTree** (p. 44).

### 4.7.3 Member Data Documentation

**4.7.3.1** `Histogram* LeafNode::m_histogram` [private]

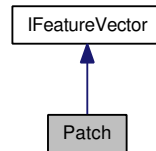
The documentation for this class was generated from the following files:

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**LeafNode.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**LeafNode.cpp**

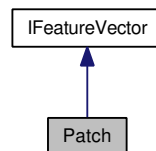
## 4.8 Patch Class Reference

```
#include <Patch.h>
```

Inheritance diagram for Patch:



Collaboration diagram for Patch:



### Public Member Functions

- **Patch** (IplImage \*image, int x, int y, int w, int h)  
*Constructor for rectangular patch.*
- **Patch** (IplImage \*image, int x, int y, int d)  
*Constructor for a square patch.*
- **Patch** (const **Patch** &other)  
*Copy constructor.*
- **~Patch** ()  
*Destructor.*
- **IFeatureVector** \* **clone** () const
- double **operator** [ ] (unsigned int k) const
- unsigned int **size** () const
- double **getValue** (int i, int j, int c) const  
*Get value of the patch at location i,j and for channel c.*
- std::pair< int, int > **getSize** () const  
*Get the geometry of the patch.*
- int **getX** () const  
*Get the x position of the patch.*
- int **getY** () const  
*Get the y position of the patch.*



- **IplImage \* getImage ()** const  
*Get the image of the patch.*
- **void setImage (IplImage \*image)**  
*Set the image.*
- **bool isValid ()** const  
*Check if a patch is totally included into its image.*
- **void toImage (IplImage \*dst)** const  
*convert the patch to an image (with given size)*
- **void saveAsImage (const char \*filename, int scale=20)** const  
*Save a patch with big scale into an image.*
- **Histogram histogram (int channel=0)** const  
*compute the color histogram of given channel (0=R)*

### Private Attributes

- **int m\_x**  
*Geometry of the patch.*
- **int m\_y**
- **int m\_w**
- **int m\_h**
- **IplImage \* m\_image**  
*Image the patch belongs to (not owned by the class patch, must be deleted somewhere else).*

## 4.8.1 Constructor & Destructor Documentation

### 4.8.1.1 Patch::Patch (IplImage \* image, int x, int y, int w, int h)

Constructor for rectangular patch.

### 4.8.1.2 Patch::Patch (IplImage \* image, int x, int y, int d)

Constructor for a square patch.

### 4.8.1.3 Patch::Patch (const Patch & other)

Copy constructor.

### 4.8.1.4 Patch::~Patch ()

Destructor.

## 4.8.2 Member Function Documentation

### 4.8.2.1 `IFeatureVector * Patch::clone () const` [virtual]

Implements `IFeatureVector` (p. 24).

### 4.8.2.2 `double Patch::operator[] (unsigned int k) const` [virtual]

Implements `IFeatureVector` (p. 24).

### 4.8.2.3 `unsigned int Patch::size () const` [virtual]

Implements `IFeatureVector` (p. 24).

### 4.8.2.4 `double Patch::getValue (int i, int j, int c) const`

Get value of the patch at location *i,j* and for channel *c*.

### 4.8.2.5 `std::pair< int, int > Patch::getSize () const`

Get the geometry of the patch.

### 4.8.2.6 `int Patch::getX () const`

Get the x position of the patch.

### 4.8.2.7 `int Patch::getY () const`

Get the y position of the patch.

### 4.8.2.8 `IplImage * Patch::getImage () const`

Get the image of the patch.

### 4.8.2.9 `void Patch::setImage (IplImage * image)`

Set the image.

### 4.8.2.10 `bool Patch::isValid () const`

Check if a patch is totally included into its image.

### 4.8.2.11 `void Patch::toImage (IplImage * dst) const`

convert the patch to an image (with given size)

**4.8.2.12 void Patch::saveAsImage (const char \**filename*, int *scale* = 20) const**

Save a patch with big scale into an image.

**4.8.2.13 Histogram Patch::histogram (int *channel* = 0) const**

compute the color histogram of given channel (0=R)

**4.8.3 Member Data Documentation****4.8.3.1 int Patch::m\_x [private]**

Geometry of the patch.

**4.8.3.2 int Patch::m\_y [private]****4.8.3.3 int Patch::m\_w [private]****4.8.3.4 int Patch::m\_h [private]****4.8.3.5 IplImage\* Patch::m\_image [private]**

Image the patch belongs to (not owned by the class patch, must be deleted somewhere else).

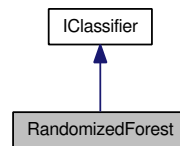
The documentation for this class was generated from the following files:

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**Patch.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**Patch.cpp**

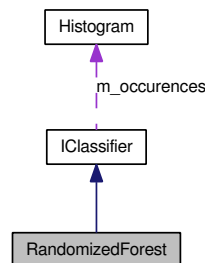
## 4.9 RandomizedForest Class Reference

```
#include <RandomizedForest.h>
```

Inheritance diagram for RandomizedForest:



Collaboration diagram for RandomizedForest:



### Public Member Functions

- **RandomizedForest** (unsigned int nbLabels=2, bool occurrence\_normalization=false, unsigned int depth=10, unsigned int nb\_trees=10, unsigned int vector\_size=15, double minValue=0., double maxValue=255.)

*Constructor.*

- **RandomizedForest** (const **RandomizedForest** &other)

*Copy constructor.*

- **RandomizedForest** & **operator=** (const **RandomizedForest** &other)

*assignement operator*

- **~RandomizedForest** ()

*Destructor.*

- void **train** (const **IFeatureVector** &vector, unsigned int label)

*training*

- **Histogram** **estimatePosterior** (const **IFeatureVector** &vector) const

*testing*

- void **load** (const std::string &filename)

*virtual method to load a classifier from a file*

- void **save** (const std::string &filename)  
*save in a file*
- **IClassifier** \* **clone** () const  
*Clone the classifier.*
- void **reset** ()  
*Clean the classifier, so that it forgets whatever it has learnt before.*
- unsigned int **getNbClasses** () const  
*Get the number of classes.*
- unsigned int **getVectorSize** () const  
*get the size of the considered square patches*
- unsigned int **getNbTrees** () const  
*Get the number of tree in the forest.*
- unsigned int **getDepth** () const  
*Get the depth of the forest.*
- double **getMinValue** () const  
*get min value expected in the feature vectors*
- double **getMaxValue** () const  
*get max value expected in the feature vectors*

### Private Member Functions

- void **clean** ()  
*clean the forest*
- void **annotate** ()  
*give ids to the nodes of the forest*

### Private Attributes

- unsigned int **m\_depth**
- unsigned int **m\_nb\_trees**
- unsigned int **m\_vector\_size**
- double **m\_min\_value**
- double **m\_max\_value**
- std::vector< **RandomizedTree** \* > **m\_trees**

## Friends

- `std::ostream & operator<< (std::ostream &out, RandomizedForest &forest)`  
*I/O for randomized forest.*
- `std::istream & operator>> (std::istream &in, RandomizedForest &forest)`

## 4.9.1 Constructor & Destructor Documentation

### 4.9.1.1 RandomizedForest::RandomizedForest (unsigned int *nbLabels* = 2, bool *occurence\_normalization* = false, unsigned int *depth* = 10, unsigned int *nb\_trees* = 10, unsigned int *vector\_size* = 15, double *minValue* = 0., double *maxValue* = 255.)

Constructor.

### 4.9.1.2 RandomizedForest::RandomizedForest (const RandomizedForest & *other*)

Copy constructor.

### 4.9.1.3 RandomizedForest::~~RandomizedForest ()

Destructor.

## 4.9.2 Member Function Documentation

### 4.9.2.1 RandomizedForest & RandomizedForest::operator= (const RandomizedForest & *other*)

assignement operator

### 4.9.2.2 void RandomizedForest::train (const IFeatureVector & *vector*, unsigned int *label*) [virtual]

training

Implements **IClassifier** (p. 21).

### 4.9.2.3 Histogram RandomizedForest::estimatePosterior (const IFeatureVector & *vector*) const [virtual]

testing

Implements **IClassifier** (p. 21).

### 4.9.2.4 void RandomizedForest::load (const std::string & *filename*) [virtual]

virtual method to load a classifier from a file

Implements **IClassifier** (p. 22).

**4.9.2.5 void RandomizedForest::save (const std::string &filename) [virtual]**

save in a file

Implements **IClassifier** (p. 22).

**4.9.2.6 IClassifier \* RandomizedForest::clone () const [virtual]**

Clone the classifier.

Implements **IClassifier** (p. 22).

**4.9.2.7 void RandomizedForest::reset () [virtual]**

Clean the classifier, so that it forgets whatever it has learnt before.

Implements **IClassifier** (p. 22).

**4.9.2.8 unsigned int RandomizedForest::getNbClasses () const**

Get the number of classes.

**4.9.2.9 unsigned int RandomizedForest::getVectorSize () const**

get the size of the considered square patches

**4.9.2.10 unsigned int RandomizedForest::getNbTrees () const**

Get the number of tree in the forest.

**4.9.2.11 unsigned int RandomizedForest::getDepth () const**

Get the depth of the forest.

**4.9.2.12 double RandomizedForest::getMinValue () const**

get min value expected in the feature vectors

**4.9.2.13 double RandomizedForest::getMaxValue () const**

get max value expected in the feature vectors

**4.9.2.14 void RandomizedForest::clean () [private]**

clean the forest

**4.9.2.15 void RandomizedForest::annotate () [private]**

give ids to the nodes of the forest

**4.9.3 Friends And Related Function Documentation****4.9.3.1 std::ostream& operator<< (std::ostream & *out*, RandomizedForest & *forest*) [friend]**

I/O for randomized forest.

**4.9.3.2 std::istream& operator>> (std::istream & *in*, RandomizedForest & *forest*) [friend]****4.9.4 Member Data Documentation****4.9.4.1 unsigned int RandomizedForest::m\_depth [private]****4.9.4.2 unsigned int RandomizedForest::m\_nb\_trees [private]****4.9.4.3 unsigned int RandomizedForest::m\_vector\_size [private]****4.9.4.4 double RandomizedForest::m\_min\_value [private]****4.9.4.5 double RandomizedForest::m\_max\_value [private]****4.9.4.6 std::vector<RandomizedTree\*> RandomizedForest::m\_trees [private]**

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

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**RandomizedForest.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**RandomizedForest.cpp**



## 4.10 RandomizedTest Class Reference

```
#include <RandomizedTest.h>
```

### Public Member Functions

- **RandomizedTest ()**  
*default constructor*
- **RandomizedTest** (unsigned int size, double minV=0., double maxV=255.)  
*Constructor of the test, the geometry is given by a patch.*
- **RandomizedTest & operator=** (const **RandomizedTest** &other)  
*Assignement operator.*
- **RandomizedTest** (const **RandomizedTest** &other)  
*Copy constructor.*
- **~RandomizedTest ()**  
*Destructor of the Randomized test.*
- bool **process** (const **IFeatureVector** &vector) const  
*Process the test on a patch.*
- unsigned int **getSize** () const  
*Get the geometry of the tested patches.*
- double **getMinValue** () const  
*get the range of the data (between 0 and range)*
- double **getMaxValue** () const

### Private Member Functions

- void **generate** ()  
*Randomly generate a test.*

### Private Attributes

- unsigned int **m\_size**  
*geometry*
- double **m\_min\_value**  
*range*
- double **m\_max\_value**
- unsigned int **m\_i**

(*random*) value of the vector to compare

- unsigned int **m\_j**
- double **m\_th**
- unsigned int **m\_type**

## Friends

- std::ostream & **operator**<< (std::ostream &out, const **RandomizedTest** &t)  
I/O.
- std::istream & **operator**>> (std::istream &in, **RandomizedTest** &t)

## 4.10.1 Constructor & Destructor Documentation

### 4.10.1.1 **RandomizedTest::RandomizedTest ()**

default constructor

### 4.10.1.2 **RandomizedTest::RandomizedTest (unsigned int *size*, double *minV* = 0 . , double *maxV* = 255 .)**

Constructor of the test, the geometry is given by a patch.

### 4.10.1.3 **RandomizedTest::RandomizedTest (const **RandomizedTest** & *other*)**

Copy constructor.

### 4.10.1.4 **RandomizedTest::~~RandomizedTest ()**

Destructor of the Randomized test.

## 4.10.2 Member Function Documentation

### 4.10.2.1 **RandomizedTest & RandomizedTest::operator= (const **RandomizedTest** & *other*)**

Assignement operator.

### 4.10.2.2 **bool RandomizedTest::process (const **IFeatureVector** & *vector*) const**

Process the test on a patch.

### 4.10.2.3 **unsigned int RandomizedTest::getSize () const**

Get the geometry of the tested patches.

#### 4.10.2.4 double RandomizedTest::getMinValue () const

get the range of the data (between 0 and range)

#### 4.10.2.5 double RandomizedTest::getMaxValue () const

#### 4.10.2.6 void RandomizedTest::generate () [private]

Randomly generate a test.

### 4.10.3 Friends And Related Function Documentation

#### 4.10.3.1 std::ostream& operator<< (std::ostream & *out*, const RandomizedTest & *t*) [friend]

I/O.

#### 4.10.3.2 std::istream& operator>> (std::istream & *in*, RandomizedTest & *t*) [friend]

### 4.10.4 Member Data Documentation

#### 4.10.4.1 unsigned int RandomizedTest::m\_size [private]

geometry

#### 4.10.4.2 double RandomizedTest::m\_min\_value [private]

range

#### 4.10.4.3 double RandomizedTest::m\_max\_value [private]

#### 4.10.4.4 unsigned int RandomizedTest::m\_i [private]

(random) value of the vector to compare

#### 4.10.4.5 unsigned int RandomizedTest::m\_j [private]

#### 4.10.4.6 double RandomizedTest::m\_th [private]

#### 4.10.4.7 unsigned int RandomizedTest::m\_type [private]

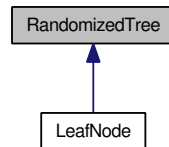
The documentation for this class was generated from the following files:

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**RandomizedTest.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**RandomizedTest.cpp**

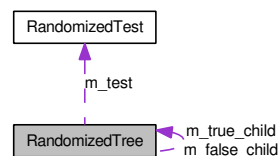
## 4.11 RandomizedTree Class Reference

```
#include <RandomizedTree.h>
```

Inheritance diagram for RandomizedTree:



Collaboration diagram for RandomizedTree:



### Public Member Functions

- **RandomizedTree** (int depth=1, int nbLabels=2, unsigned int vector\_size=1, double minV=0., double maxV=255.)

*Constructor.*

- **RandomizedTree** (int depth, int nbLabels, const **RandomizedTest** &test)

*constructor given a test*

- **RandomizedTree** (const **RandomizedTree** &other)

*Copy Constructor.*

- virtual void **copy** (**RandomizedTree** \*other)

*copy a randomized tree*

- virtual ~**RandomizedTree** ()

*Destructor.*

- void **train** (const **IFeatureVector** &vector, int label)

*Process patch.*

- virtual bool **isLeaf** () const

*Tell if the node is a leaf.*

- void **annotate** (int &current\_id)

*Put Id to the nodes of the tree.*

- int **getNumberNodes** () const

*Compute the number of nodes of the tree.*

- void **compact** (std::vector< **RandomizedTree** \* > &nodes, std::vector< **t\_Edge** > &edges) const  
*sum up the tree into two lists : nodes and edges*
- void **expand** (std::map< int, **RandomizedTree** \* > &nodes, std::vector< **t\_Edge** > &edges)  
*create a tree from the node list and the edges list*
- void **toDot** (const char \*filename)  
*plot the tree as a graphViz .dot file*
- int **getID** () const  
*Getters.*
- int **getDepth** () const
- int **getNbClasses** () const
- unsigned int **getVectorSize** () const
- double **getMinValue** () const
- double **getMaxValue** () const
- const **LeafNode** \* **patchToLeaf** (const **IFeatureVector** &vector) const  
*Turn a patch into a **LeafNode** (p. 25) for testing purposes.*

## Protected Member Functions

- virtual void **train** (const **IFeatureVector** &patch, int label, int d)

## Protected Attributes

- int **m\_depth**  
*The depth of the tree.*
- int **m\_nb\_classes**  
*the number of classes considered in the classification*
- unsigned int **m\_vector\_size**  
*size of the feature vector*
- double **m\_min\_value**  
*min value of the data*
- double **m\_max\_value**  
*max value of the data*
- **RandomizedTest** \* **m\_test**  
*a test, build randomly at the construction of the tree*
- int **m\_id**

## Private Attributes

- **RandomizedTree \* m\_true\_child**  
*the True Child and the False child -> remember it's a decision tree...*
- **RandomizedTree \* m\_false\_child**

## Friends

- **std::ostream & operator<<** (std::ostream &out, const **RandomizedTree** &tree)  
*I/O for trees.*
- **std::istream & operator>>** (std::istream &in, **RandomizedTree** &tree)

## 4.11.1 Constructor & Destructor Documentation

### 4.11.1.1 **RandomizedTree::RandomizedTree** (int *depth* = 1, int *nbLabels* = 2, unsigned int *vector\_size* = 1, double *minV* = 0., double *maxV* = 255.)

Constructor.

### 4.11.1.2 **RandomizedTree::RandomizedTree** (int *depth*, int *nbLabels*, const **RandomizedTest** & *test*)

constructor given a test

### 4.11.1.3 **RandomizedTree::RandomizedTree** (const **RandomizedTree** & *other*)

Copy Constructor.

### 4.11.1.4 **RandomizedTree::~~RandomizedTree** () [virtual]

Destructor.

## 4.11.2 Member Function Documentation

### 4.11.2.1 **void RandomizedTree::copy** (**RandomizedTree** \* *other*) [virtual]

copy a randomized tree

Reimplemented in **LeafNode** (p. 26).

### 4.11.2.2 **void RandomizedTree::train** (const **IFeatureVector** & *vector*, int *label*)

Process patch.

**4.11.2.3 bool RandomizedTree::isLeaf () const** [virtual]

Tell if the node is a leaf.

Reimplemented in **LeafNode** (p. 26).

**4.11.2.4 void RandomizedTree::annotate (int & current\_id)**

Put Id to the nodes of the tree.

**4.11.2.5 int RandomizedTree::getNumberNodes () const**

Compute the number of nodes of the tree.

**4.11.2.6 void RandomizedTree::compact (std::vector< RandomizedTree \* > & nodes, std::vector< t\_Edge > & edges) const**

sum up the tree into two lists : nodes and edges

**4.11.2.7 void RandomizedTree::expand (std::map< int, RandomizedTree \* > & nodes, std::vector< t\_Edge > & edges)**

create a tree from the node list and the edges list

**4.11.2.8 void RandomizedTree::toDot (const char \* filename)**

plot the tree as a graphViz .dot file

**4.11.2.9 int RandomizedTree::getID () const**

Getters.

**4.11.2.10 int RandomizedTree::getDepth () const****4.11.2.11 int RandomizedTree::getNbClasses () const****4.11.2.12 unsigned int RandomizedTree::getVectorSize () const****4.11.2.13 double RandomizedTree::getMinValue () const****4.11.2.14 double RandomizedTree::getMaxValue () const****4.11.2.15 const LeafNode \* RandomizedTree::patchToLeaf (const IFeatureVector & vector) const**

Turn a patch into a **LeafNode** (p. 25) for testing purposes.

return an outlier leaf

return an outlier leaf

**4.11.2.16** `void RandomizedTree::train (const IFeatureVector & patch, int label, int d)`  
 [protected, virtual]

Reimplemented in **LeafNode** (p. 27).

### 4.11.3 Friends And Related Function Documentation

**4.11.3.1** `std::ostream& operator<< (std::ostream & out, const RandomizedTree & tree)`  
 [friend]

I/O for trees.

**4.11.3.2** `std::istream& operator>> (std::istream & in, RandomizedTree & tree)` [friend]

### 4.11.4 Member Data Documentation

**4.11.4.1** `int RandomizedTree::m_depth` [protected]

The depth of the tree.

**4.11.4.2** `int RandomizedTree::m_nb_classes` [protected]

the number of classes considered in the classification

**4.11.4.3** `unsigned int RandomizedTree::m_vector_size` [protected]

size of the feature vector

**4.11.4.4** `double RandomizedTree::m_min_value` [protected]

min value of the data

**4.11.4.5** `double RandomizedTree::m_max_value` [protected]

max value of the data

**4.11.4.6** `RandomizedTest* RandomizedTree::m_test` [protected]

a test, build randomly at the construction of the tree

**4.11.4.7** `int RandomizedTree::m_id` [protected]

**4.11.4.8** `RandomizedTree* RandomizedTree::m_true_child` [private]

the True Child and the False child -> remember it's a decision tree...



**4.11.4.9 RandomizedTree\* RandomizedTree::m\_false\_child** [private]

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

- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/**RandomizedTree.h**
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/**RandomizedTree.cpp**



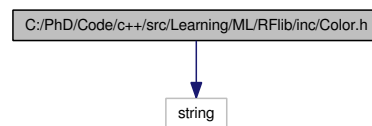
## Chapter 5

# RFlib File Documentation

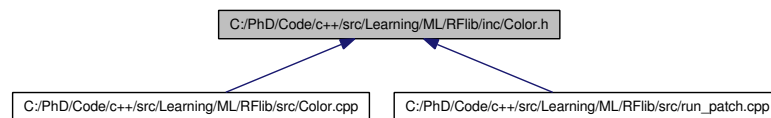
### 5.1 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Color.h File Reference

```
#include <string>
```

Include dependency graph for Color.h:



This graph shows which files directly or indirectly include this file:



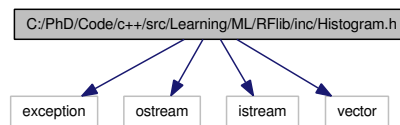
### Classes

- class **Color**

## 5.2 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Histogram.h File Reference

```
#include <exception>
#include <ostream>
#include <istream>
#include <vector>
```

Include dependency graph for Histogram.h:



This graph shows which files directly or indirectly include this file:



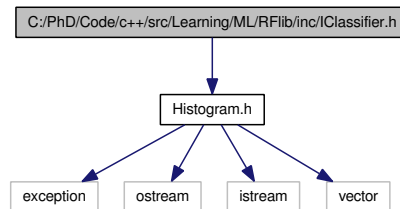
### Classes

- class **Histogram**
- class **HistogramOperationError**
- class **HistogramAssigmentError**

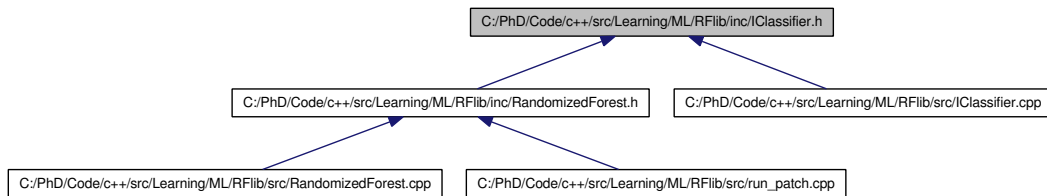
## 5.3 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IClassifier.h File Reference

```
#include "Histogram.h"
```

Include dependency graph for IClassifier.h:



This graph shows which files directly or indirectly include this file:



## Classes

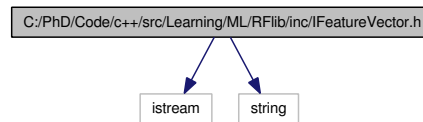
- class **IClassifier**

## 5.4 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IFeatureVector.h File Reference

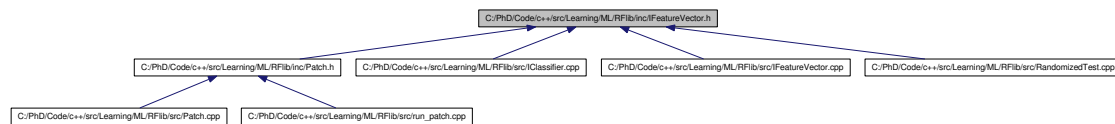
```
#include <istream>
```

```
#include <string>
```

Include dependency graph for IFeatureVector.h:



This graph shows which files directly or indirectly include this file:



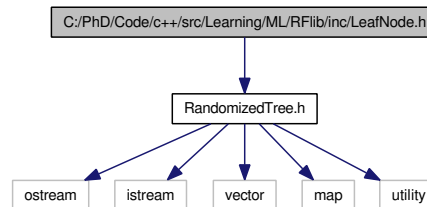
## Classes

- class **IFeatureVector**

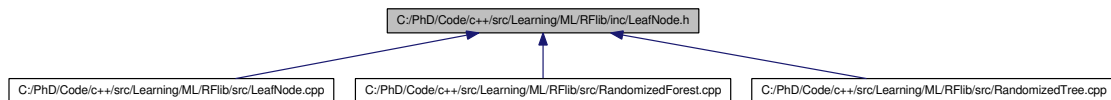
## 5.5 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/LeafNode.h File Reference

```
#include "RandomizedTree.h"
```

Include dependency graph for LeafNode.h:



This graph shows which files directly or indirectly include this file:



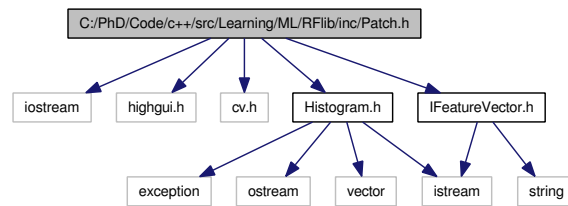
### Classes

- class **LeafNode**

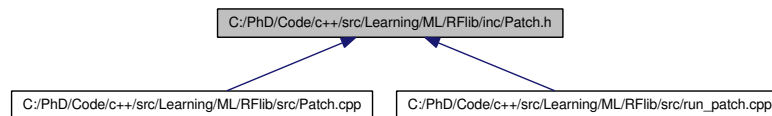
## 5.6 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h File Reference

```
#include <iostream>
#include <highgui.h>
#include <cv.h>
#include "Histogram.h"
#include "IFeatureVector.h"
```

Include dependency graph for Patch.h:



This graph shows which files directly or indirectly include this file:



## Classes

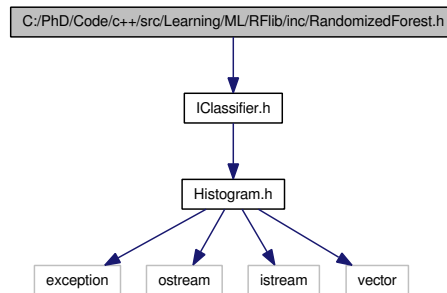
- class **Patch**



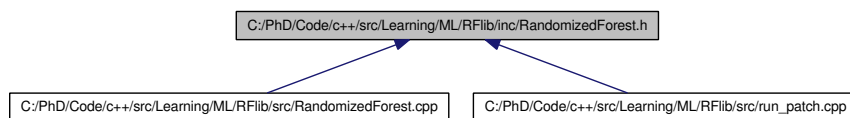
## 5.7 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h File Reference

```
#include "IClassifier.h"
```

Include dependency graph for RandomizedForest.h:



This graph shows which files directly or indirectly include this file:

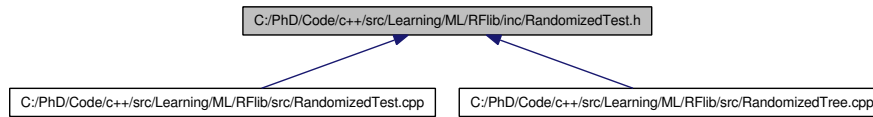


### Classes

- class **RandomizedForest**

## 5.8 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTest.h File Reference

This graph shows which files directly or indirectly include this file:



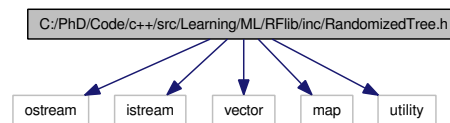
### Classes

- class **RandomizedTest**

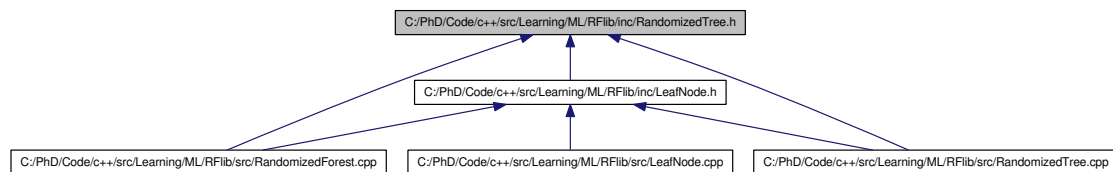
## 5.9 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTree.h File Reference

```
#include <ostream>
#include <istream>
#include <vector>
#include <map>
#include <utility>
```

Include dependency graph for RandomizedTree.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **RandomizedTree**

### Typedefs

- typedef std::pair< std::pair< int, int >, int > **t\_Edge**

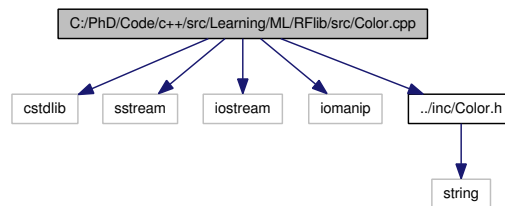
#### 5.9.1 Typedef Documentation

##### 5.9.1.1 typedef std::pair<std::pair<int,int>,int> t\_Edge

## 5.10 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Color.cpp File Reference

```
#include <cstdlib>
#include <sstream>
#include <iostream>
#include <iomanip>
#include "../inc/Color.h"
```

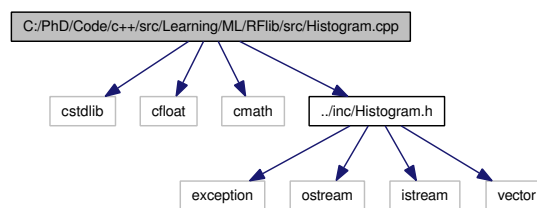
Include dependency graph for Color.cpp:



## 5.11 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Histogram.cpp File Reference

```
#include <cstdlib>
#include <cfloat>
#include <cmath>
#include "../inc/Histogram.h"
```

Include dependency graph for Histogram.cpp:



### Functions

- std::ostream & **operator**<< (std::ostream &out, const **Histogram** &hist)
- std::istream & **operator**>> (std::istream &in, **Histogram** &hist)

#### 5.11.1 Function Documentation

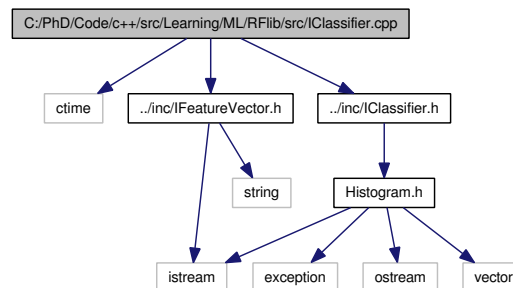
5.11.1.1 std::ostream& operator<< (std::ostream & *out*, const Histogram & *hist*)

5.11.1.2 std::istream& operator>> (std::istream & *in*, Histogram & *hist*)

## 5.12 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IClassifier.cpp File Reference

```
#include <ctime>
#include "../inc/IFeatureVector.h"
#include "../inc/IClassifier.h"
```

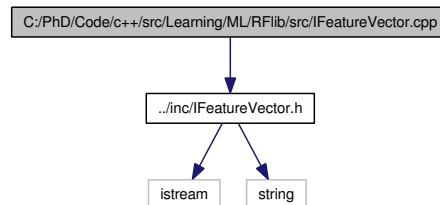
Include dependency graph for IClassifier.cpp:



## 5.13 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IFeatureVector.cpp File Reference

```
#include "../inc/IFeatureVector.h"
```

Include dependency graph for IFeatureVector.cpp:

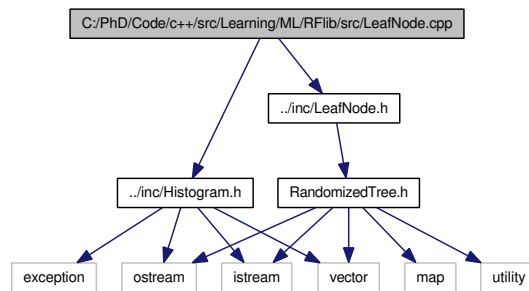


## 5.14 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/LeafNode.cpp File Reference

```
#include "../inc/Histogram.h"
```

```
#include "../inc/LeafNode.h"
```

Include dependency graph for LeafNode.cpp:



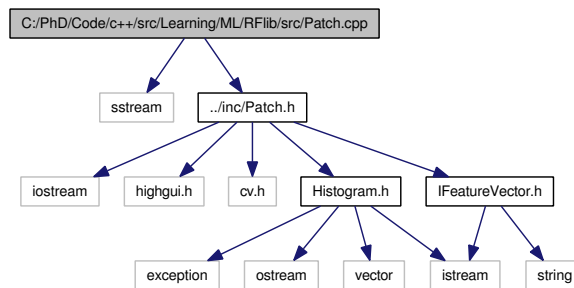


## 5.15 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Patch.cpp File Reference

```
#include <sstream>
```

```
#include "../inc/Patch.h"
```

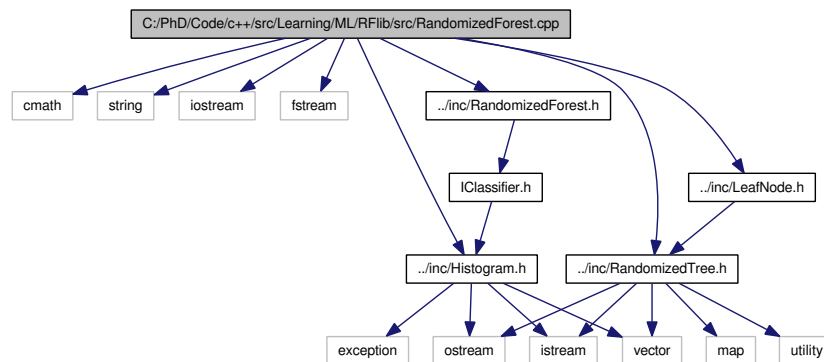
Include dependency graph for Patch.cpp:



## 5.16 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedForest.cpp File Reference

```
#include <cmath>
#include <string>
#include <iostream>
#include <fstream>
#include "../inc/Histogram.h"
#include "../inc/RandomizedTree.h"
#include "../inc/LeafNode.h"
#include "../inc/RandomizedForest.h"
```

Include dependency graph for RandomizedForest.cpp:



### Functions

- `std::ostream & operator<< (std::ostream &out, RandomizedForest &forest)`
- `std::istream & operator>> (std::istream &in, RandomizedForest &forest)`

#### 5.16.1 Function Documentation

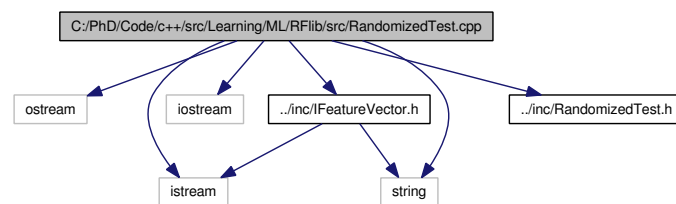
5.16.1.1 `std::ostream& operator<< (std::ostream & out, RandomizedForest & forest)`

5.16.1.2 `std::istream& operator>> (std::istream & in, RandomizedForest & forest)`

## 5.17 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTest.cpp File Reference

```
#include <ostream>
#include <istream>
#include <iostream>
#include <string>
#include "../inc/IFeatureVector.h"
#include "../inc/RandomizedTest.h"
```

Include dependency graph for RandomizedTest.cpp:



### Functions

- `std::ostream & operator<< (std::ostream &out, const RandomizedTest &t)`
- `std::istream & operator>> (std::istream &in, RandomizedTest &t)`

#### 5.17.1 Function Documentation

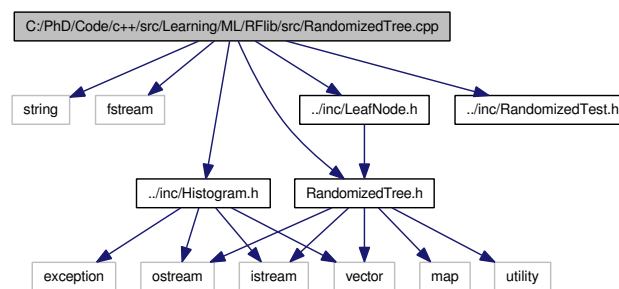
5.17.1.1 `std::ostream& operator<< (std::ostream & out, const RandomizedTest & t)`

5.17.1.2 `std::istream& operator>> (std::istream & in, RandomizedTest & t)`

## 5.18 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTree.cpp File Reference

```
#include <string>
#include <fstream>
#include "../inc/Histogram.h"
#include "../inc/RandomizedTest.h"
#include "../inc/LeafNode.h"
#include "../inc/RandomizedTree.h"
```

Include dependency graph for RandomizedTree.cpp:



### Functions

- `std::ostream & operator<< (std::ostream &out, const RandomizedTree &v)`
- `std::istream & operator>> (std::istream &in, RandomizedTree &tree)`

### 5.18.1 Function Documentation

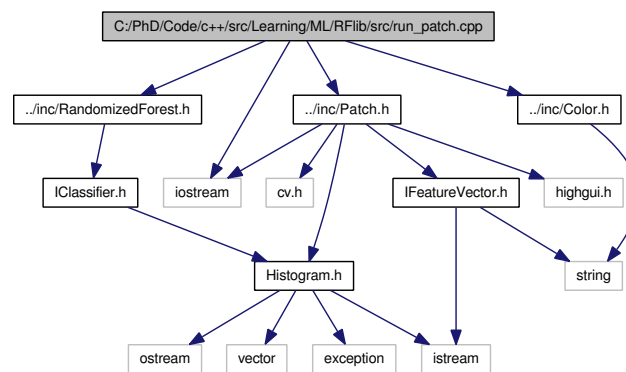
**5.18.1.1** `std::ostream& operator<< (std::ostream & out, const RandomizedTree & v)`

**5.18.1.2** `std::istream& operator>> (std::istream & in, RandomizedTree & tree)`

## 5.19 C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run\_patch.cpp File Reference

```
#include <iostream>
#include "../inc/RandomizedForest.h"
#include "../inc/Color.h"
#include "../inc/Patch.h"
```

Include dependency graph for run\_patch.cpp:



### Functions

- `int main (int argc, char **argv)`

#### 5.19.1 Function Documentation

##### 5.19.1.1 `int main (int argc, char ** argv)`

# Index

- ~Color
  - Color, 9
- ~Histogram
  - Histogram, 13
- ~IClassifier
  - IClassifier, 21
- ~IFeatureVector
  - IFeatureVector, 24
- ~LeafNode
  - LeafNode, 26
- ~Patch
  - Patch, 29
- ~RandomizedForest
  - RandomizedForest, 34
- ~RandomizedTest
  - RandomizedTest, 38
- ~RandomizedTree
  - RandomizedTree, 42
- annotate
  - RandomizedForest, 35
  - RandomizedTree, 43
- argMax
  - Histogram, 14
- argMin
  - Histogram, 14
- basis
  - Histogram, 14
- battacharyya
  - Histogram, 15
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Color.h, 47
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Histogram.h, 48
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IClassifier.h, 49
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IFeatureVector.h, 50
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/LeafNode.h, 51
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h, 52
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h, 53
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTest.h, 54
- C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTree.h, 55
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Color.cpp, 56
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Histogram.cpp, 57
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IClassifier.cpp, 58
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IFeatureVector.cpp, 59
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/LeafNode.cpp, 60
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Patch.cpp, 61
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedForest.cpp, 62
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTest.cpp, 63
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTree.cpp, 64
- C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run\_patch.cpp, 65
- classify
  - IClassifier, 22
- clean
  - RandomizedForest, 35
- clone
  - IClassifier, 22
  - IFeatureVector, 24
  - Patch, 30
  - RandomizedForest, 35
- color, 7
- ~Color, 9
- Color, 8, 9
- getAlpha, 9
- getB, 9
- getG, 9
- getR, 9
- hsv2rgb, 10
- isDefault, 10
- m\_alpha, 10
- ForBst, 10
- m\_G, 10

- m\_R, 10
- operator!=, 9
- operator=, 9
- operator==, 9
- rainbow, 9
- rainbow\_cube, 10
- random, 10
- rgb2hsv, 10
- setAlpha, 10
- setB, 10
- setG, 10
- setR, 10
- toString, 10
- compact
  - RandomizedTree, 43
- copy
  - LeafNode, 26
  - RandomizedTree, 42
- distanceL1
  - Histogram, 15
- distanceL2
  - Histogram, 15
- distanceLP
  - Histogram, 15
- dot
  - Histogram, 16
- egreedy
  - Histogram, 16
- estimatePosterior
  - IClassifier, 21
  - RandomizedForest, 34
- expand
  - RandomizedTree, 43
- fill
  - Histogram, 14
- generate
  - RandomizedTest, 39
- getAlpha
  - Color, 9
- getB
  - Color, 9
- getDepth
  - RandomizedForest, 35
  - RandomizedTree, 43
- getG
  - Color, 9
- getHistogram
  - LeafNode, 26
- getID
  - RandomizedTree, 43
- getImage
  - Patch, 30
- getMaxValue
  - RandomizedForest, 35
  - RandomizedTest, 39
  - RandomizedTree, 43
- getMinValue
  - RandomizedForest, 35
  - RandomizedTest, 38
  - RandomizedTree, 43
- getNbClasses
  - Histogram, 14
  - RandomizedForest, 35
  - RandomizedTree, 43
- getNbLabels
  - IClassifier, 22
- getNbTrees
  - RandomizedForest, 35
- getNumberNodes
  - RandomizedTree, 43
- getR
  - Color, 9
- getSize
  - Patch, 30
  - RandomizedTest, 38
- getValue
  - Patch, 30
- getVectorSize
  - RandomizedForest, 35
  - RandomizedTree, 43
- getWeight
  - Histogram, 16
- getX
  - Patch, 30
- getY
  - Patch, 30
- hellinger
  - Histogram, 15
- Histogram, 11
  - ~Histogram, 13
  - argMax, 14
  - argMin, 14
  - basis, 14
  - battacharyya, 15
  - distanceL1, 15
  - distanceL2, 15
  - distanceLP, 15
  - dot, 16
  - egreedy, 16
  - fill, 14
  - getNbClasses, 14
  - getWeight, 16
  - hellinger, 15

- Histogram, 13
- kl, 15
- linearCombination, 15
- m\_data, 16
- m\_nb\_classes, 16
- m\_weight, 17
- maxValue, 14
- minValue, 14
- norm, 14
- normalize, 15
- normalize2, 15
- ones, 14
- operator \*, 15
- operator \*=, 15
- operator <<, 16
- operator >>, 16
- operator+, 15
- operator+=, 15
- operator/, 15
- operator/=: 15
- operator=, 14
- operator[], 14
- resetWeight, 16
- resize, 16
- sample, 16
- scale, 15
- setWeight, 16
- sum, 14
- zeros, 14
- histogram
  - Patch, 31
- Histogram.cpp
  - operator <<, 57
  - operator >>, 57
- HistogramAssignmentError, 18
- HistogramOperationError, 19
- hsv2rgb
  - Color, 10
- IClassifier, 20
  - ~IClassifier, 21
  - classify, 22
  - clone, 22
  - estimatePosterior, 21
  - getNbLabels, 22
  - IClassifier, 21
  - load, 21
  - m\_nb\_labels, 22
  - m\_occurence\_normalization, 22
  - m\_occurrences, 22
  - reset, 22
  - save, 22
  - setOccurenceNormalization, 22
  - train, 21
- IFeatureVector, 24
  - ~IFeatureVector, 24
  - clone, 24
  - IFeatureVector, 24
  - operator[], 24
  - size, 24
- isDefault
  - Color, 10
- isLeaf
  - LeafNode, 26
  - RandomizedTree, 42
- isValid
  - Patch, 30
- kl
  - Histogram, 15
- LeafNode, 25
  - ~LeafNode, 26
  - copy, 26
  - getHistogram, 26
  - isLeaf, 26
  - LeafNode, 26
  - m\_histogram, 27
  - operator[], 26
  - train, 26
- linearCombination
  - Histogram, 15
- load
  - IClassifier, 21
  - RandomizedForest, 34
- m\_alpha
  - Color, 10
- m\_B
  - Color, 10
- m\_data
  - Histogram, 16
- m\_depth
  - RandomizedForest, 36
  - RandomizedTree, 44
- m\_false\_child
  - RandomizedTree, 44
- m\_G
  - Color, 10
- m\_h
  - Patch, 31
- m\_histogram
  - LeafNode, 27
- m\_i
  - RandomizedTest, 39
- m\_id
  - RandomizedTree, 44
- m\_image



- Patch, 31
- m\_j
  - RandomizedTest, 39
- m\_max\_value
  - RandomizedForest, 36
  - RandomizedTest, 39
  - RandomizedTree, 44
- m\_min\_value
  - RandomizedForest, 36
  - RandomizedTest, 39
  - RandomizedTree, 44
- m\_nb\_classes
  - Histogram, 16
  - RandomizedTree, 44
- m\_nb\_labels
  - IClassifier, 22
- m\_nb\_trees
  - RandomizedForest, 36
- m\_occurence\_normalization
  - IClassifier, 22
- m\_occurrences
  - IClassifier, 22
- m\_R
  - Color, 10
- m\_size
  - RandomizedTest, 39
- m\_test
  - RandomizedTree, 44
- m\_th
  - RandomizedTest, 39
- m\_trees
  - RandomizedForest, 36
- m\_true\_child
  - RandomizedTree, 44
- m\_type
  - RandomizedTest, 39
- m\_vector\_size
  - RandomizedForest, 36
  - RandomizedTree, 44
- m\_w
  - Patch, 31
- m\_weight
  - Histogram, 17
- m\_x
  - Patch, 31
- m\_y
  - Patch, 31
- main
  - run\_patch.cpp, 65
- maxValue
  - Histogram, 14
- minValue
  - Histogram, 14
- norm
  - Histogram, 14
- normalize
  - Histogram, 15
- normalize2
  - Histogram, 15
- ones
  - Histogram, 14
- operator \*
  - Histogram, 15
- operator \*=
  - Histogram, 15
- operator !=
  - Color, 9
- operator <<
  - Histogram, 16
  - Histogram.cpp, 57
  - RandomizedForest, 36
  - RandomizedForest.cpp, 62
  - RandomizedTest, 39
  - RandomizedTest.cpp, 63
  - RandomizedTree, 44
  - RandomizedTree.cpp, 64
- operator >>
  - Histogram, 16
  - Histogram.cpp, 57
  - RandomizedForest, 36
  - RandomizedForest.cpp, 62
  - RandomizedTest, 39
  - RandomizedTest.cpp, 63
  - RandomizedTree, 44
  - RandomizedTree.cpp, 64
- operator +
  - Histogram, 15
- operator +=
  - Histogram, 15
- operator /
  - Histogram, 15
- operator /=
  - Histogram, 15
- operator =
  - Color, 9
  - Histogram, 14
  - RandomizedForest, 34
  - RandomizedTest, 38
- operator ==
  - Color, 9
- operator []
  - Histogram, 14
  - IFeatureVector, 24
  - LeafNode, 26
  - Patch, 30

- Patch, 28
  - ~Patch, 29
  - clone, 30
  - getImage, 30
  - getSize, 30
  - getValue, 30
  - getX, 30
  - getY, 30
  - histogram, 31
  - isValid, 30
  - m\_h, 31
  - m\_image, 31
  - m\_w, 31
  - m\_x, 31
  - m\_y, 31
  - operator[], 30
  - Patch, 29
  - saveAsImage, 30
  - setImage, 30
  - size, 30
  - toImage, 30
- patchToLeaf
  - RandomizedTree, 43
- process
  - RandomizedTest, 38
- rainbow
  - Color, 9
- rainbow\_cube
  - Color, 10
- random
  - Color, 10
- RandomizedForest, 32
  - ~RandomizedForest, 34
  - annotate, 35
  - clean, 35
  - clone, 35
  - estimatePosterior, 34
  - getDepth, 35
  - getMaxValue, 35
  - getMinValue, 35
  - getNbClasses, 35
  - getNbTrees, 35
  - getVectorSize, 35
  - load, 34
  - m\_depth, 36
  - m\_max\_value, 36
  - m\_min\_value, 36
  - m\_nb\_trees, 36
  - m\_trees, 36
  - m\_vector\_size, 36
  - operator<<, 36
  - operator>>, 36
  - operator=, 34
  - RandomizedForest, 34
  - reset, 35
  - save, 34
  - train, 34
- RandomizedForest.cpp
  - operator<<, 62
  - operator>>, 62
- RandomizedTest, 37
  - ~RandomizedTest, 38
  - generate, 39
  - getMaxValue, 39
  - getMinValue, 38
  - getSize, 38
  - m\_i, 39
  - m\_j, 39
  - m\_max\_value, 39
  - m\_min\_value, 39
  - m\_size, 39
  - m\_th, 39
  - m\_type, 39
  - operator<<, 39
  - operator>>, 39
  - operator=, 38
  - process, 38
  - RandomizedTest, 38
- RandomizedTest.cpp
  - operator<<, 63
  - operator>>, 63
- RandomizedTree, 40
  - ~RandomizedTree, 42
  - annotate, 43
  - compact, 43
  - copy, 42
  - expand, 43
  - getDepth, 43
  - getID, 43
  - getMaxValue, 43
  - getMinValue, 43
  - getNbClasses, 43
  - getNumberNodes, 43
  - getVectorSize, 43
  - isLeaf, 42
  - m\_depth, 44
  - m\_false\_child, 44
  - m\_id, 44
  - m\_max\_value, 44
  - m\_min\_value, 44
  - m\_nb\_classes, 44
  - m\_test, 44
  - m\_true\_child, 44
  - m\_vector\_size, 44
  - operator<<, 44
  - operator>>, 44
  - patchToLeaf, 43

- RandomizedTree, 42
  - toDot, 43
  - train, 42, 43
- RandomizedTree.cpp
  - operator<<, 64
  - operator>>, 64
- RandomizedTree.h
  - t\_Edge, 55
- reset
  - IClassifier, 22
  - RandomizedForest, 35
- resetWeight
  - Histogram, 16
- resize
  - Histogram, 16
- rgb2hsv
  - Color, 10
- run\_patch.cpp
  - main, 65
- sample
  - Histogram, 16
- save
  - IClassifier, 22
  - RandomizedForest, 34
- saveAsImage
  - Patch, 30
- scale
  - Histogram, 15
- setAlpha
  - Color, 10
- setB
  - Color, 10
- setG
  - Color, 10
- setImage
  - Patch, 30
- setOccurenceNormalization
  - IClassifier, 22
- setR
  - Color, 10
- setWeight
  - Histogram, 16
- size
  - IFeatureVector, 24
  - Patch, 30
- sum
  - Histogram, 14
- t\_Edge
  - RandomizedTree.h, 55
- toDot
  - RandomizedTree, 43
- toImage
  - Patch, 30
- toString
  - Color, 10
- train
  - IClassifier, 21
  - LeafNode, 26
  - RandomizedForest, 34
  - RandomizedTree, 42, 43
- zeros
  - Histogram, 14