RFlib Reference Manual

1.0

Generated by Doxygen 1.5.4

Tue Mar 2 17:27:59 2010

Contents

Chapter 1

RFlib Hierarchical Index

1.1 RFlib Class Hierarchy

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

Chapter 2

RFlib Class Index

2.1 RFlib Class List

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

Chapter 3

RFlib File Index

3.1 RFlib File List

Here is a list of all files with brief descriptions:

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)

  Mutliplication element-wise of two histgrams.

- **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 historgram 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*)

Assignement 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 histgrams.

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 historgram 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 HistogramAssignementError 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>
```

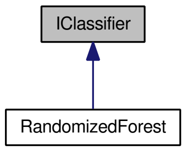
The documentation for this class was generated from the following file:

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

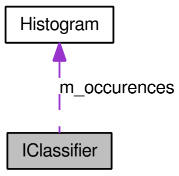
4.5 IClassifier Class Reference

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

Inheritance diagram for IClassifier:



Collaboration diagram for IClassifier:



Public Member Functions

- **IClassifier** (unsigned int nbLabels=2, bool occurence_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 ?

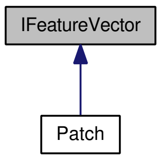
The documentation for this class was generated from the following files:

- 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).

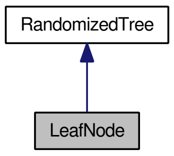
The documentation for this class was generated from the following files:

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

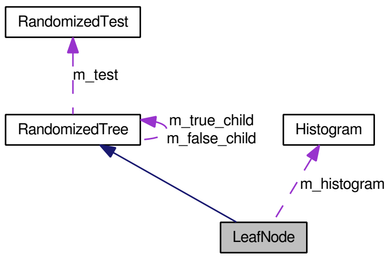
4.7 LeafNode Class Reference

`#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::$\sim$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 ith bin of the histogram.

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

the same version, but that allows to modify the ith 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]`

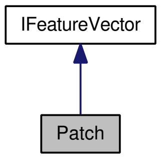
The documentation for this class was generated from the following files:

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

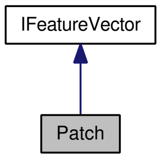
4.8 Patch Class Reference

`#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.

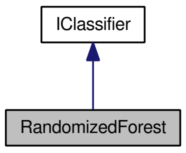
The documentation for this class was generated from the following files:

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

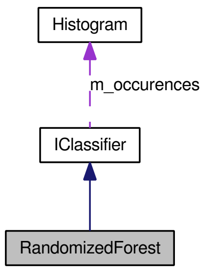
4.9 RandomizedForest Class Reference

`#include <RandomizedForest.h>`

Inheritance diagram for RandomizedForest:



Collaboration diagram for RandomizedForest:



Public Member Functions

- **RandomizedForest** (unsigned int nbLabels=2, bool occurence_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 patchs.

- 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 patchs.

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]`

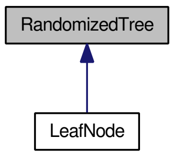
The documentation for this class was generated from the following files:

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

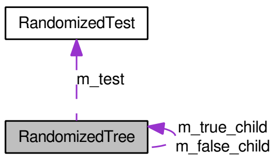
4.11 RandomizedTree Class Reference

`#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**

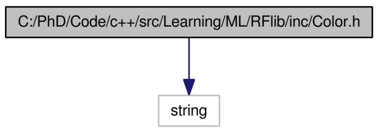
Chapter 5

RFlib File Documentation

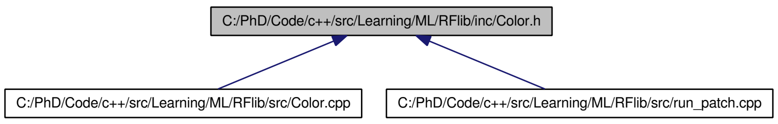
5.1 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Color.h File Reference

```
#include <string>
```

Include dependency graph for Color.h:



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


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

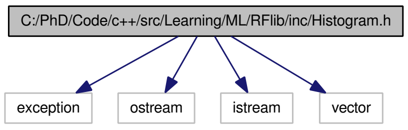
Classes

- class **Color**

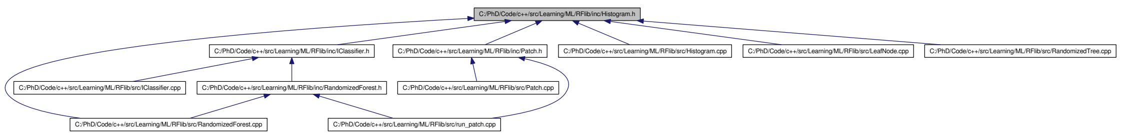
5.2 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Histogram.h File Reference

```
#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:


C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Histogram.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IClassifier.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Histogram.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/LeafNode.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTree.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IClassifier.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Patch.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedForest.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run_patch.cpp

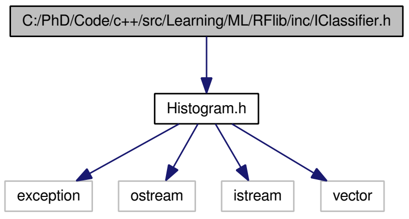
Classes

- class **Histogram**
- class **HistogramOperationError**
- class **HistogramAssignementError**

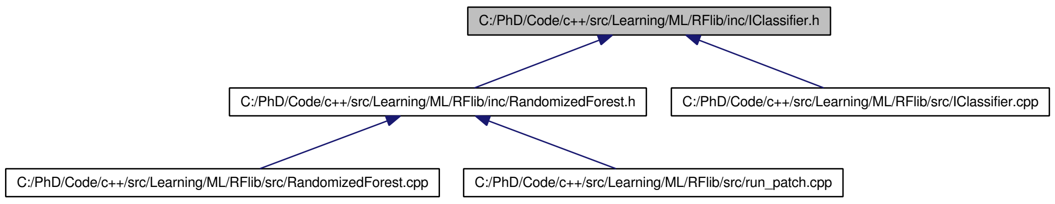
5.3 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IClassifier.h File Reference

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

Include dependency graph for IClassifier.h:



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


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

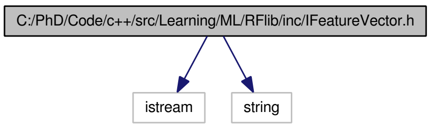
Classes

- class **IClassifier**

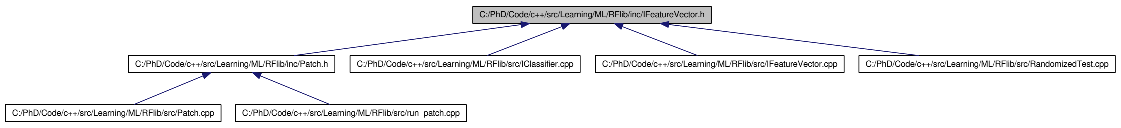
5.4 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IFeatureVector.h File Reference

```
#include <istream>
#include <string>
```

Include dependency graph for IFeatureVector.h:



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


C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/IFeatureVector.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IClassifier.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/IFeatureVector.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedTest.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/Patch.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run_patch.cpp

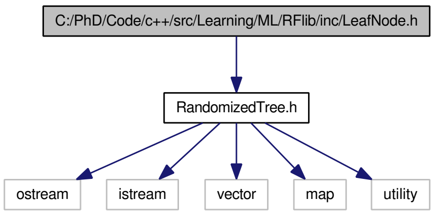
Classes

- class **IFeatureVector**

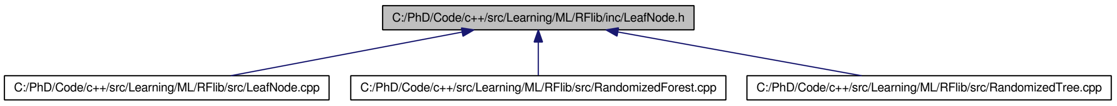
5.5 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/LeafNode.h File Reference

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

Include dependency graph for LeafNode.h:



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


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

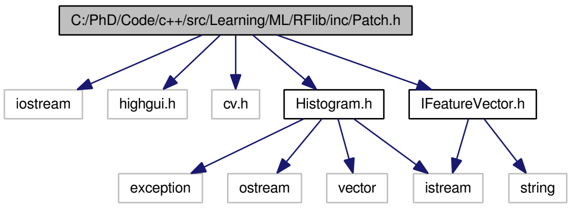
Classes

- class **LeafNode**

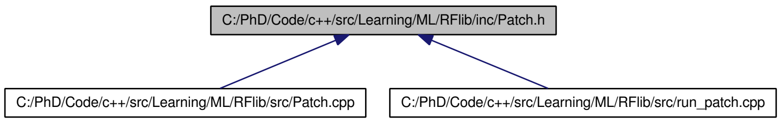
5.6 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/Patch.h File Reference

```
#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:


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

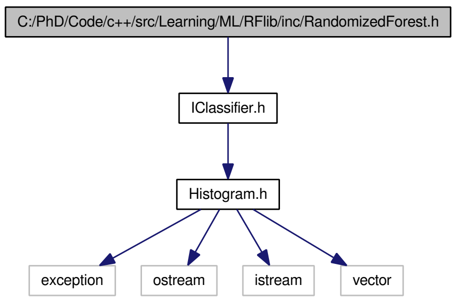
Classes

- class **Patch**

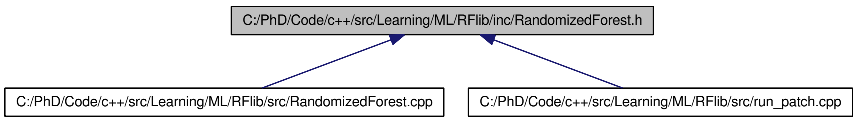
5.7 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h File Reference

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

Include dependency graph for RandomizedForest.h:


C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h
IClassifier.h
Histogram.h
exception
ostream
istream
vector

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

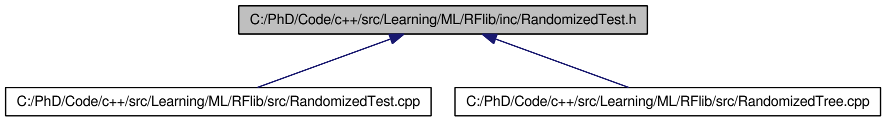

C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedForest.h
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/RandomizedForest.cpp
C:/PhD/Code/c++/src/Learning/ML/RFlib/src/run_patch.cpp

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:


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

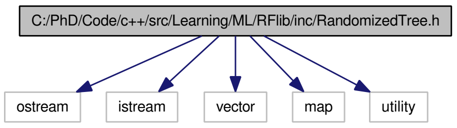
Classes

- class **RandomizedTest**

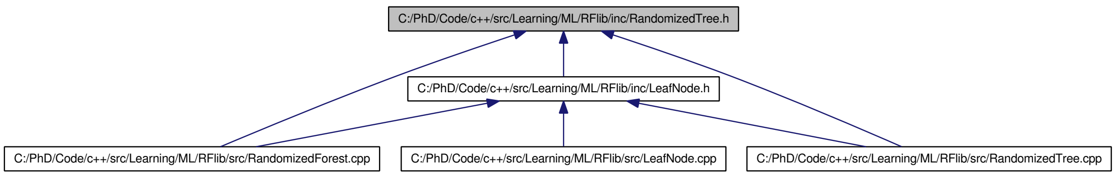
5.9 C:/PhD/Code/c++/src/Learning/ML/RFlib/inc/RandomizedTree.h File Reference

```
#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:


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

Classes

- class **RandomizedTree**

Typedefs

- typedef std::pair< std::pair< int, int >, int > **t_Edge**

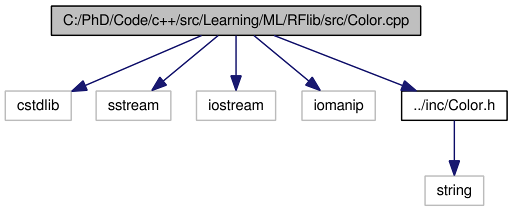
5.9.1 Typedef Documentation

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"
```

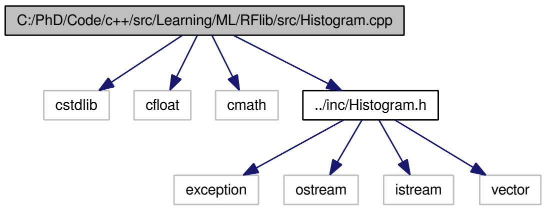
Include dependency graph for Color.cpp:



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

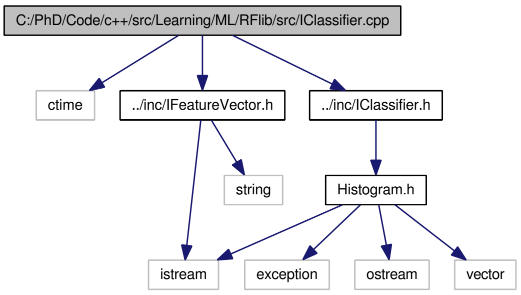
5.11.1.1 std::ostream& operator<< (std::ostream & *out*, const Histogram & *hist*)

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"
```

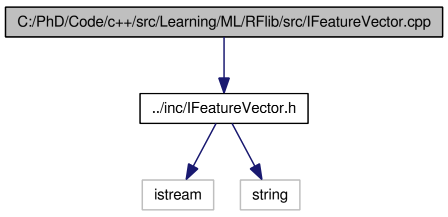
Include dependency graph for IClassifier.cpp:



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

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

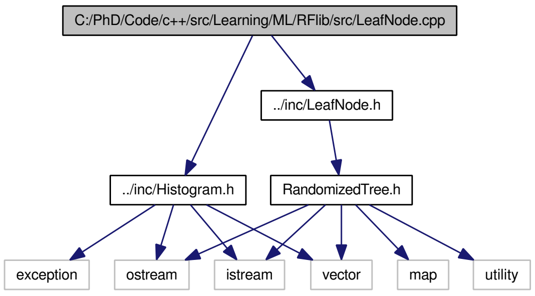
Include dependency graph for IFeatureVector.cpp:


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

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

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

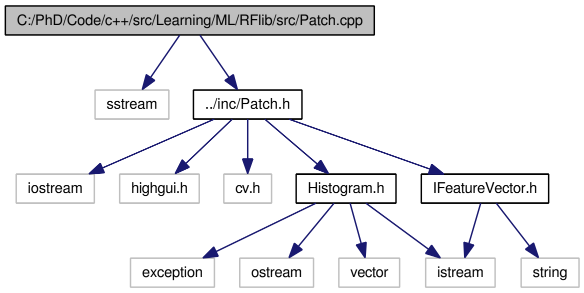
Include dependency graph for LeafNode.cpp:



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

```
#include <sstream>
#include "../inc/Patch.h"
```

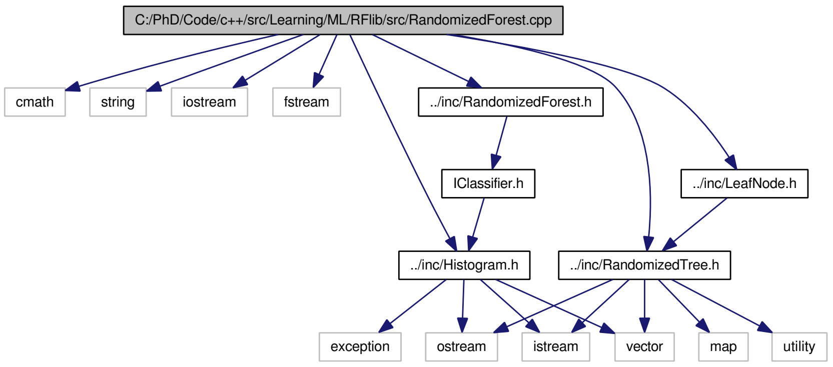
Include dependency graph for Patch.cpp:



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

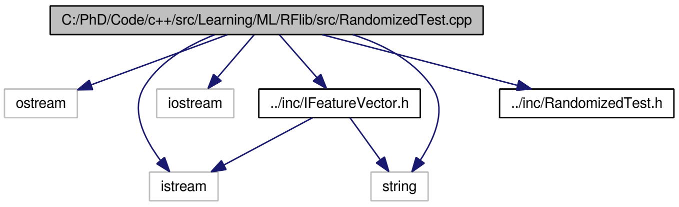
5.16.1.1 std::ostream& operator<< (std::ostream & *out*, RandomizedForest & *forest*)

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

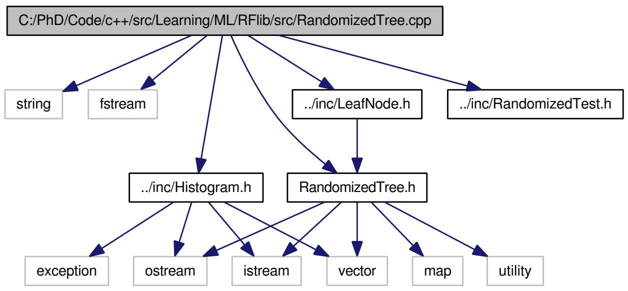
5.17.1.1 std::ostream& operator<< (std::ostream & *out*, const RandomizedTest & *t*)

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

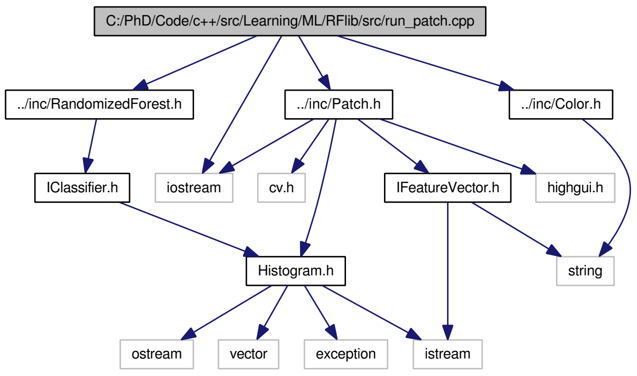
5.18.1.1 std::ostream& operator<< (std::ostream & *out*, const RandomizedTree & *v*)

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