Classification Algorithms System

V0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

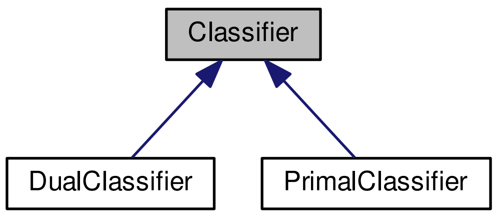
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Classifier Class Reference

Inheritance diagram for Classifier:



Public Member Functions

- virtual void train ()=0

  Function that execute the training phase of a classification algorithm.
- virtual int evaluate (Point x)=0

  Returns the class of a feature point based on the trained classifier.

4.1.1 Member Function Documentation

4.1.1.1 evaluate()

```
virtual int Classifier::evaluate (
            Point x )  [pure virtual]
```

Returns the class of a feature point based on the trained classifier.

Parameters

Point	x (???) Features point to be evaluated.

Returns

int

4.1.1.2 train()

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

Function that execute the training phase of a classification algorithm.

Returns

void

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

- includes/Classifier.hpp

4.2 CrossValidation Class Reference

Public Member Functions

- **CrossValidation** (Data sample, Classifier classifier)
- double **kFold** (int fold, int seed)
- void **validation** (int fold, int qtde)
- Data **getTestSample** ()
- Data **getTrainSample** ()

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

- includes/CrossValidation.hpp

4.3 Data Class Reference

Wrapper for the dataset data.

```
#include <Data.hpp>
```

Public Member Functions

- Data ()

  Constructor for empty data.
- Data (std::string dataset)

  Data constructor to load a dataset from a file.
- int getSize ()

  Returns the size of the dataset.
- int getDim ()

  Returns the dimension of the dataset.
- Point getPoint (int index)

  Returns the point with the given index.
- std::vector< Point > getPoints ()

  Returns the vector of Points of the sample.
- std::vector< int > getFeaturesNames ()

  Returns the features names.
- Statistics getStatistics ()

  Returns a class with the statistics info of the sample.
- int getNumberPositivePoints ()

  Return the number of positive points.
- int getNumberNegativePoints ()

  Return the number of negative points.
- bool isEmpty ()

  Returns if there's a dataset loaded.
- bool load (std::string file)

  Load a dataset from a file.
- Data copy ()

  Returns a copy of the data.
- bool insertPoint (Data sample, int id)

  Insert a point to the data from another sample.
- bool insertPoint (Point p)

  Insert a point to the end of vector points.
- std::vector< bool > removePoints (std::vector< int > ids)

  Remove several points from the sample.
- bool removePoint (int pid)

  Remove a point from the data.
- bool removeFeatures (std::vector< int > feats)

  Remove several features from the sample.
- void changeXVector (std::vector< int > index)

  Change the x vector of a sample.
- void **operator=** (const Data &)

4.3.1 Detailed Description

Wrapper for the dataset data.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Data()

```
Data::Data (
             std::string dataset )
```

Data constructor to load a dataset from a file.

Parameters

dataset	(???) Path to the dataset to be loaded.

4.3.3 Member Function Documentation

4.3.3.1 changeXVector()

```
void Data::changeXVector (
             std::vector< int > index )
```

Change the x vector of a sample.

Parameters

index	(???) Indexes of the change to be made.

Returns

void

4.3.3.2 copy()

```
Data Data::copy ( )
```

Returns a copy of the data.

Returns

Data

4.3.3.3 getDim()

```
int Data::getDim ( )
```

Returns the dimension of the dataset.

Returns

int

4.3.3.4 getFeaturesNames()

```
vector< int > Data::getFeaturesNames ( )
```

Returns the features names.

Returns

std::vector<int>

4.3.3.5 getNumberNegativePoints()

```
int Data::getNumberNegativePoints ( )
```

Return the number of negative points.

Returns

int

4.3.3.6 getNumberPositivePoints()

```
int Data::getNumberPositivePoints ( )
```

Return the number of positive points.

Returns

int

4.3.3.7 getPoint()

```
Point Data::getPoint (
            int index )
```

Returns the point with the given index.

Parameters

index	Position of a point in the points array.

Returns

std::vector<Points>

4.3.3.8 getPoints()

```
vector< Point > Data::getPoints ( )
```

Returns the vector of Points of the sample.

Returns

std::vector<Points>

4.3.3.9 getSize()

```
int Data::getSize ( )
```

Returns the size of the dataset.

Returns

int

4.3.3.10 getStatistics()

```
Statistics Data::getStatistics ( )
```

Returns a class with the statistics info of the sample.

Returns

Statistics

4.3.3.11 insertPoint() [1/2]

```
bool Data::insertPoint (
            Data sample,
            int id )
```

Insert a point to the data from another sample.

Parameters

sample	(???) Sample with the point to be added.
id	(???) Index of the point to be added.

Returns

bool

4.3.3.12 insertPoint() [2/2]

```
bool Data::insertPoint (
             Point p )
```

Insert a point to the end of vector points.

Parameters

p	(???) Point to be inserted.

Returns

bool

4.3.3.13 isEmpty()

```
bool Data::isEmpty ( )
```

Returns if there's a dataset loaded.

Returns

bool

4.3.3.14 load()

```
bool Data::load (
             std::string file )
```

Load a dataset from a file.

Parameters

file	(???) Path to dataset file.

Returns

bool

4.3.3.15 removeFeatures()

```
bool Data::removeFeatures (
             std::vector< int > feats )
```

Remove several features from the sample.

Parameters

feats	(???) Names of the features to be removed (must be sorted).

Returns

boolean informing if all features were succesfully removed.

4.3.3.16 removePoint()

```
bool Data::removePoint (
             int pid )
```

Remove a point from the data.

Parameters

pid	(???) Index of the point to be removed.

Returns

bool

4.3.3.17 removePoints()

```
vector< bool > Data::removePoints (
             std::vector< int > ids )
```

Remove several points from the sample.

Parameters

ids	(???) Ids of the points to be removed (must be sorted).

Returns

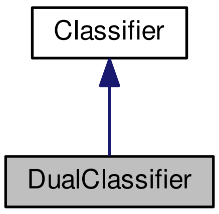
booleans informing which points were removed succesfully.

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

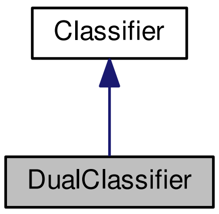
- includes/Data.hpp
- src/Data.cpp

4.4 DualClassifier Class Reference

Inheritance diagram for DualClassifier:



Collaboration diagram for DualClassifier:



Additional Inherited Members

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

- includes/DualClassifier.hpp

4.5 FeatureSelection Class Reference

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

- includes/FeatureSelection.hpp

4.6 Kernel Class Reference

Class for the kernel computations.

```
#include <Kernel.hpp>
```

4.6.1 Detailed Description

Class for the kernel computations.

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

- includes/Kernel.hpp

4.7 Point Class Reference

Class of a Point of doubles in a space of n dimensions.

```
#include <Point.hpp>
```

Public Member Functions

- **Point** (int dim)
- double dot (std::vector< double > p)

  Computes the dot product with a vector.
- double norm (int p=2)

  Returns the p-norm of the point.

Public Attributes

- std::vector< double > x

  Features values.
- double y = 0

  Point classification.
- int id = 0

  Point identification.

Friends

- std::ostream & **operator**<< (std::ostream &output, const Point &p)

4.7.1 Detailed Description

Class of a Point of doubles in a space of n dimensions.

4.7.2 Member Function Documentation

4.7.2.1 dot()

```
double Point::dot (
              std::vector< double > p )
```

Computes the dot product with a vector.

Parameters

p	(???)

Returns

double

4.7.2.2 norm()

```
double Point::norm (
              int p = 2 )
```

Returns the p-norm of the point.

Parameters

p	(???) p of the norm (euclidean norm is the default).

Returns

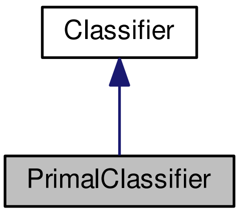
double

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

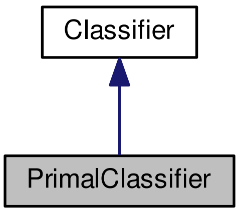
- includes/Point.hpp
- src/Point.cpp

4.8 PrimalClassifier Class Reference

Inheritance diagram for PrimalClassifier:



Collaboration diagram for PrimalClassifier:



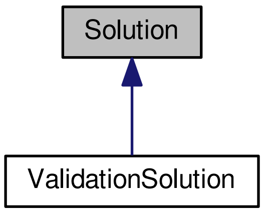
Additional Inherited Members

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

- includes/PrimalClassifier.hpp

4.9 Solution Class Reference

Inheritance diagram for Solution:



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

- includes/Solution.hpp

4.10 Statistics Class Reference

Class with methods for statistical computations.

```
#include <Statistics.hpp>
```

Static Public Member Functions

- static double mean (std::vector< double > p)

  Compute the mean (average) of a vector.
- static double getFeatureMean (Data data, int index)

  Computes the mean of a feature in the sample.
- static double variance (std::vector< double > p)

  Compute the variance of a vector.
- static double variance (Data data, int index)

  Compute the variance of a sample.
- static double stdev (std::vector< double > p)

  Compute the standard deviation of a vector.
- static double getFeatureStdev (Data data, int index)

  Computes the standard deviation of a feature.

Friends

- class **Data**

4.10.1 Detailed Description

Class with methods for statistical computations.

4.10.2 Member Function Documentation

4.10.2.1 getFeatureMean()

```
double Statistics::getFeatureMean (
            Data data,
            int index )  [static]
```

Computes the mean of a feature in the sample.

Parameters

data	(???) Sample where the feature is located.
index	(???) Index of the feature to compute the mean.

Returns

double

4.10.2.2 getFeatureStdev()

```
double Statistics::getFeatureStdev (
            Data data,
            int index )  [static]
```

Computes the standard deviation of a feature.

Parameters

data	(???) Sample where the feature is located.
index	(???) Index of teh feature to compute the standard deviation.

Returns

double

4.10.2.3 mean()

```
double Statistics::mean (
            std::vector< double > p )  [static]
```

Compute the mean (average) of a vector.

Parameters

p	(???) Point to compute the mean.

Returns

double

4.10.2.4 stdev()

```
double Statistics::stdev (
            std::vector< double > p )  [static]
```

Compute the standard deviation of a vector.

Parameters

p	(???) Point to compute stdev.

Returns

double

4.10.2.5 variance() [1/2]

```
static double Statistics::variance (
            std::vector< double > p )  [static]
```

Compute the variance of a vector.

Parameters

p	(???) Vector to compute the variance.

Returns

double

4.10.2.6 variance() [2/2]

```
double Statistics::variance (
            Data data,
            int index )  [static]
```

Compute the variance of a sample.

Parameters

data	(???) Sample to compute the variance.
index	(???) Index of the feature to be ignored. (-1 dont ignore any feature)

Returns

double

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

- includes/Statistics.hpp
- src/Statistics.cpp

4.11 Validation Class Reference

Class of methods for the validation of ML algorithms.

```
#include <Validation.hpp>
```

Public Member Functions

- **CrossValidation** (Data sample, Classifier classifier)
- double **kFold** (int fold, int seed)
- void **validation** (int fold, int qtde)
- Data **getTestSample** ()
- Data **getTrainSample** ()

4.11.1 Detailed Description

Class of methods for the validation of ML algorithms.

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

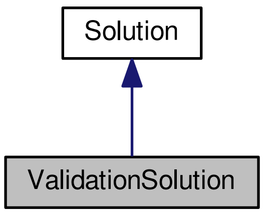
- includes/Validation.hpp

4.12 ValidationSolution Class Reference

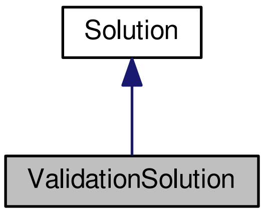
Solution for the validation of a ML method.

```
#include <ValidationSolution.hpp>
```

Inheritance diagram for ValidationSolution:



Collaboration diagram for ValidationSolution:



4.12.1 Detailed Description

Solution for the validation of a ML method.

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

- includes/ValidationSolution.hpp

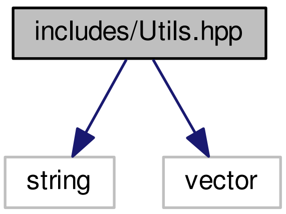
Chapter 5

File Documentation

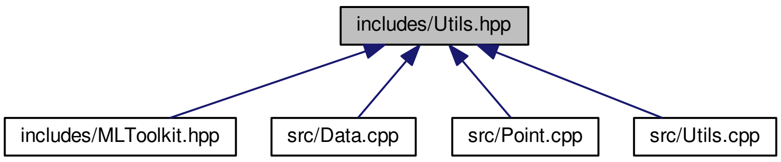
5.1 includes/Utils.hpp File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for Utils.hpp:



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



Macros

- #define **INF** 1E8

Enumerations

- enum **NormType** { **NORM_LINF** = 0, **NORM_L1** = 1, **NORM_L2** = 2 }

Functions

- bool is_number (std::string str)

    Verify if the string is a number.
- int stoin (std::string str)

    Converts the string to an integer.
- double stodn (std::string str)

    Converts the string to a double.
- double maxAbsElement (std::vector< double > x)

    Returns the max absolute element.

5.1.1 Detailed Description

Utils functions

Author

Mateus Coutinho Marim

5.1.2 Function Documentation

5.1.2.1 is_number()

```
bool is_number (
              std::string str )
```

Verify if the string is a number.

Parameters

str	String to be tested.

Returns

bool

5.1.2.2 maxAbsElement()

```
double maxAbsElement (
              std::vector< double > x )
```

Returns the max absolute element.

Parameters

x	The vector used to obtain the max element.

Returns

The max absolute element found.

5.1.2.3 stodn()

```
double stodn (
              std::string str )
```

Converts the string to a double.

Parameters

str	The string to be converted.

Returns

The double resulted from the conversion.

5.1.2.4 stoin()

```
int stoin (
              std::string str )
```

Converts the string to an integer.

Parameters

str	String to be converted.

Returns

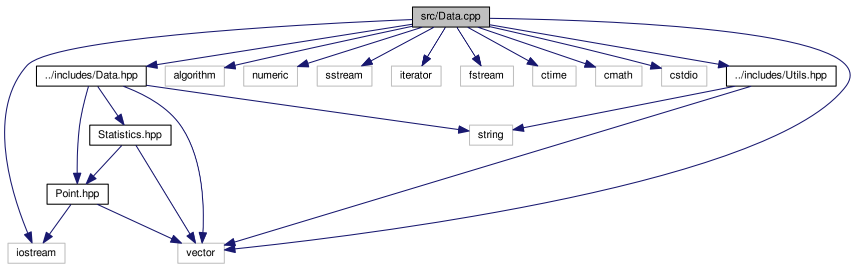
The integer resulted from the conversion.

5.2 src/Data.cpp File Reference

Implementation of the Data class methods.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <sstream>
#include <iterator>
#include <fstream>
#include <ctime>
#include <cmath>
#include <cstdio>
#include "../includes/Data.hpp"
#include "../includes/Utils.hpp"
```

Include dependency graph for Data.cpp:



5.2.1 Detailed Description

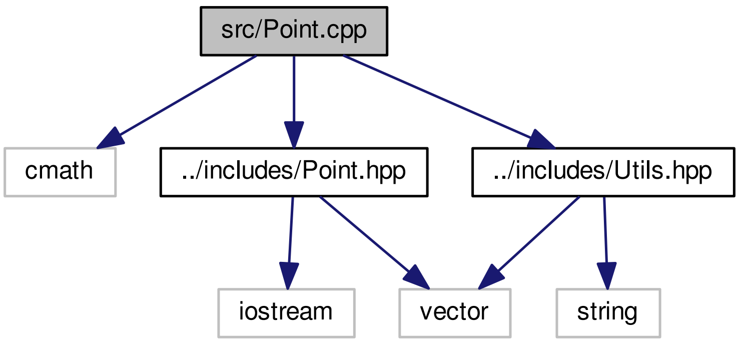
Implementation of the Data class methods.

5.3 src/Point.cpp File Reference

Implementation of the Point class methods.

```
#include <cmath>
#include "../includes/Point.hpp"
#include "../includes/Utils.hpp"
```

Include dependency graph for Point.cpp:



Functions

- ostream & **operator**<< (ostream &output, const Point &p)

5.3.1 Detailed Description

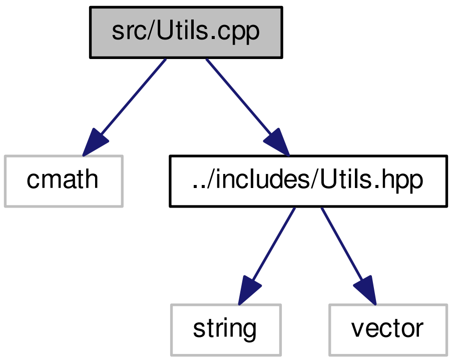
Implementation of the Point class methods.

5.4 src/Utils.cpp File Reference

Implementation of methods for general use in the system.

```
#include <cmath>
#include "../includes/Utils.hpp"
```

Include dependency graph for Utils.cpp:



Functions

- bool **is_number** (string str)
- int **stoin** (string str)
- double **stodn** (string str)
- double **maxAbsElement** (vector< double > x)

5.4.1 Detailed Description

Implementation of methods for general use in the system.

Utils functions

Author

Mateus Coutinho Marim

Index