

# Classification Algorithms System

## V0.1

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	Classifier Class Reference . . . . .	7
4.1.1	Member Function Documentation . . . . .	7
4.1.1.1	evaluate() . . . . .	7
4.1.1.2	train() . . . . .	8
4.2	Data Class Reference . . . . .	8
4.2.1	Detailed Description . . . . .	10
4.2.2	Constructor & Destructor Documentation . . . . .	10
4.2.2.1	Data() [1/2] . . . . .	10
4.2.2.2	Data() [2/2] . . . . .	10
4.2.3	Member Function Documentation . . . . .	10
4.2.3.1	changeXVector() . . . . .	10
4.2.3.2	copy() . . . . .	12
4.2.3.3	copyZero() . . . . .	12
4.2.3.4	getDim() . . . . .	12
4.2.3.5	getFeaturesNames() . . . . .	13

4.2.3.6	<a href="#">getIndex()</a>	13
4.2.3.7	<a href="#">getNumberNegativePoints()</a>	13
4.2.3.8	<a href="#">getNumberPositivePoints()</a>	13
4.2.3.9	<a href="#">getPoint()</a>	13
4.2.3.10	<a href="#">getPoints()</a>	14
4.2.3.11	<a href="#">getSize()</a>	14
4.2.3.12	<a href="#">getStatistics()</a>	14
4.2.3.13	<a href="#">insertPoint()</a> [1/2]	14
4.2.3.14	<a href="#">insertPoint()</a> [2/2]	15
4.2.3.15	<a href="#">isEmpty()</a>	15
4.2.3.16	<a href="#">isNormalized()</a>	15
4.2.3.17	<a href="#">join()</a>	16
4.2.3.18	<a href="#">load()</a>	16
4.2.3.19	<a href="#">normalize()</a> [1/2]	16
4.2.3.20	<a href="#">normalize()</a> [2/2]	17
4.2.3.21	<a href="#">removeFeatures()</a>	17
4.2.3.22	<a href="#">removePoint()</a>	17
4.2.3.23	<a href="#">removePoints()</a>	18
4.2.3.24	<a href="#">setClasses()</a>	18
4.3	<a href="#">DualClassifier Class Reference</a>	19
4.4	<a href="#">FeatureSelection Class Reference</a>	19
4.5	<a href="#">Gnuplot Class Reference</a>	20
4.5.1	<a href="#">Member Function Documentation</a>	22
4.5.1.1	<a href="#">is_valid()</a>	22
4.5.1.2	<a href="#">operator&lt;&lt;()</a>	22
4.5.1.3	<a href="#">plot_equation()</a>	23
4.5.1.4	<a href="#">plot_equation3d()</a>	23
4.5.1.5	<a href="#">plot_image()</a>	23
4.5.1.6	<a href="#">plot_x()</a>	24
4.5.1.7	<a href="#">plot_xy()</a>	24

4.5.1.8	<a href="#">plot_xy_err()</a>	24
4.5.1.9	<a href="#">plotfile_x()</a>	24
4.5.1.10	<a href="#">plotfile_xy()</a>	24
4.5.1.11	<a href="#">plotfile_xy_err()</a>	25
4.5.1.12	<a href="#">plotfile_xyz()</a>	25
4.5.1.13	<a href="#">replot()</a>	25
4.5.1.14	<a href="#">set_contour()</a>	25
4.5.1.15	<a href="#">set_GNUPlotPath()</a>	25
4.5.1.16	<a href="#">set_hidden3d()</a>	26
4.5.1.17	<a href="#">set_legend()</a>	26
4.5.1.18	<a href="#">set_multiplot()</a>	26
4.5.1.19	<a href="#">set_smooth()</a>	27
4.5.1.20	<a href="#">set_style()</a>	27
4.5.1.21	<a href="#">set_surface()</a>	27
4.5.1.22	<a href="#">set_terminal_std()</a>	27
4.5.1.23	<a href="#">set_title()</a>	28
4.5.1.24	<a href="#">set_xautoscale()</a>	28
4.5.1.25	<a href="#">set_yautoscale()</a>	28
4.5.1.26	<a href="#">set_zautoscale()</a>	29
4.5.1.27	<a href="#">unset_contour()</a>	29
4.5.1.28	<a href="#">unset_hidden3d()</a>	29
4.5.1.29	<a href="#">unset_legend()</a>	30
4.5.1.30	<a href="#">unset_multiplot()</a>	30
4.5.1.31	<a href="#">unset_smooth()</a>	30
4.5.1.32	<a href="#">unset_surface()</a>	31
4.5.1.33	<a href="#">unset_title()</a>	31
4.5.1.34	<a href="#">unset_xlogscale()</a>	31
4.5.1.35	<a href="#">unset_ylogscale()</a>	32
4.5.1.36	<a href="#">unset_zlogscale()</a>	32
4.6	<a href="#">GnuplotException Class Reference</a>	32

4.6.1	Detailed Description	33
4.7	Kernel Class Reference	34
4.7.1	Detailed Description	34
4.7.2	Member Function Documentation	34
4.7.2.1	norm()	34
4.8	Point Class Reference	35
4.8.1	Detailed Description	35
4.8.2	Member Function Documentation	35
4.8.2.1	dot()	35
4.8.2.2	norm()	36
4.9	PrimalClassifier Class Reference	36
4.10	Solution Class Reference	37
4.11	Statistics Class Reference	37
4.11.1	Detailed Description	38
4.11.2	Member Function Documentation	38
4.11.2.1	getDistCenters()	38
4.11.2.2	getDistCentersWithoutFeats()	39
4.11.2.3	getFeatureMean()	39
4.11.2.4	getFeatureStdev()	39
4.11.2.5	getRadius()	40
4.11.2.6	mean()	40
4.11.2.7	stdev()	41
4.11.2.8	variance() [1/2]	41
4.11.2.9	variance() [2/2]	41
4.12	Validation Class Reference	42
4.12.1	Detailed Description	42
4.12.2	Member Function Documentation	42
4.12.2.1	partTrainTest()	42
4.13	ValidationSolution Class Reference	43
4.13.1	Detailed Description	43
4.14	Visualisation Class Reference	44
4.14.1	Detailed Description	44
4.14.2	Member Function Documentation	44
4.14.2.1	plot2D()	44
4.14.2.2	plot3D()	45
4.14.2.3	setSample()	45
4.14.2.4	setStyle()	45
4.14.2.5	setTitle()	46

<b>5 File Documentation</b>	<b>47</b>
5.1 includes/Utils.hpp File Reference . . . . .	47
5.1.1 Detailed Description . . . . .	48
5.1.2 Function Documentation . . . . .	48
5.1.2.1 is_number() . . . . .	48
5.1.2.2 maxAbsElement() . . . . .	49
5.1.2.3 stodn() . . . . .	49
5.1.2.4 stoin() . . . . .	49
5.2 src/Data.cpp File Reference . . . . .	50
5.2.1 Detailed Description . . . . .	50
5.3 src/Point.cpp File Reference . . . . .	51
5.3.1 Detailed Description . . . . .	51
5.4 src/Utils.cpp File Reference . . . . .	51
5.4.1 Detailed Description . . . . .	52
<b>Index</b>	<b>53</b>





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Classifier . . . . .	7
DualClassifier . . . . .	19
PrimalClassifier . . . . .	36
Data . . . . .	8
FeatureSelection . . . . .	19
Gnuplot . . . . .	20
Kernel . . . . .	34
Point . . . . .	35
runtime_error	
GnuplotException . . . . .	32
Solution . . . . .	37
ValidationSolution . . . . .	43
Statistics . . . . .	37
Validation . . . . .	42
Visualisation . . . . .	44



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Classifier</a>	7
<a href="#">Data</a>	
Wrapper for the dataset data	8
<a href="#">DualClassifier</a>	19
<a href="#">FeatureSelection</a>	19
<a href="#">Gnuplot</a>	20
<a href="#">GnuplotException</a>	
A C++ interface to gnuplot	32
<a href="#">Kernel</a>	
Class for the kernel computations	34
<a href="#">Point</a>	
Class of a <a href="#">Point</a> of doubles in a space of n dimensions	35
<a href="#">PrimalClassifier</a>	36
<a href="#">Solution</a>	37
<a href="#">Statistics</a>	
Class with methods for statistical computations	37
<a href="#">Validation</a>	
Class of methods for the validation of ML algorithms	42
<a href="#">ValidationSolution</a>	
<a href="#">Solution</a> for the validation of a ML method	43
<a href="#">Visualisation</a>	
Class for visualize data using gnuplot	44



## Chapter 3

# File Index

### 3.1 File List

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

includes/ <b>Classifier.hpp</b>	??
includes/ <b>Data.hpp</b>	??
includes/ <b>DualClassifier.hpp</b>	??
includes/ <b>FeatureSelection.hpp</b>	??
includes/ <b>gnuplot_i.hpp</b>	??
includes/ <b>Kernel.hpp</b>	??
includes/ <b>MLToolkit.hpp</b>	??
includes/ <b>Point.hpp</b>	??
includes/ <b>PrimalClassifier.hpp</b>	??
includes/ <b>Solution.hpp</b>	??
includes/ <b>Statistics.hpp</b>	??
includes/ <b>Utils.hpp</b>	47
includes/ <b>Validation.hpp</b>	??
includes/ <b>ValidationSolution.hpp</b>	??
includes/ <b>Visualisation.hpp</b>	??
src/ <b>Data.cpp</b>	
Implementation of the <b>Data</b> class methods	50
src/ <b>Point.cpp</b>	
Implementation of the <b>Point</b> class methods	51
src/ <b>Utils.cpp</b>	
Implementation of methods for general use in the system	51

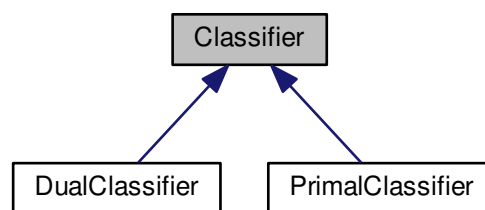


## Chapter 4

# 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**

<a href="#">Point</a>	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 Data Class Reference**

Wrapper for the dataset data.

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

**Public Member Functions**

- [Data](#) (const char \*pos\_class="1", const char \*neg\_class="-1")  
*Constructor for empty data.*
- [Data](#) (std::string dataset, const char \*pos\_class="1", const char \*neg\_class="-1")  
*Data constructor to load a dataset from a file.*
- void **write** (std::string fname, std::string ext)
- 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.*
- void **setPoint** (int index, [Point](#) p)
- 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.*
- `std::vector< int >` [getIndex](#) ()  
*Returns the vector of indexes.*
- `int` [getNumberPositivePoints](#) ()  
*Return the number of positive points.*
- `int` [getNumberNegativePoints](#) ()  
*Return the number of negative points.*
- `void` [setClasses](#) (`std::string` pos, `std::string` neg)  
*setClasses Set the classes of the dataset.*
- `bool` [isEmpty](#) ()  
*Returns if there's a dataset loaded.*
- `bool` [isNormalized](#) ()  
*Returns if the dataset is normalized.*
- `bool` [load](#) (`std::string` file)  
*Load a dataset from a file.*
- [Data](#) [copy](#) ()  
*Returns a copy of the data.*
- [Data](#) [copyZero](#) ()  
*Returns a copy of the data with zero points.*
- `void` [join](#) ([Data](#) data)  
*Merge one dataset with another.*
- `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` [normalize](#) (`double` p=2)  
*normalize Normalize the dataset using a Lp-norm.*
- `void` **operator=** (`const` [Data](#) &)
- `void` **clear** ()

### Static Public Member Functions

- `static void` [normalize](#) (`std::vector< double >` &p, `double` q)  
*normalize Normalize a vector using a Lp-norm.*

### Friends

- `std::ostream &` **operator<<** (`std::ostream &`output, `const` [Data](#) &data)

### 4.2.1 Detailed Description

Wrapper for the dataset data.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Data() [1/2]

```
Data::Data (
    const char * pos_class = "1",
    const char * neg_class = "-1" )
```

Constructor for empty data.

##### Parameters

<i>pos_class</i>	String representing the positive class on the dataset.
<i>neg_class</i>	String representing the negative class on the dataset.

#### 4.2.2.2 Data() [2/2]

```
Data::Data (
    std::string dataset,
    const char * pos_class = "1",
    const char * neg_class = "-1" )
```

[Data](#) constructor to load a dataset from a file.

##### Parameters

<i>dataset</i>	(???) Path to the dataset to be loaded.
<i>pos_class</i>	String representing the positive class on the dataset.
<i>neg_class</i>	String representing the negative class on the dataset.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 changeXVector()

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

Change the x vector of a sample.

**Parameters**

<i>index</i>	(???) Indexes of the change to be made.
--------------	---

**Returns**

void

**4.2.3.2 copy()**

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

Returns a copy of the data.

**Returns**

Data

**4.2.3.3 copyZero()**

```
Data Data::copyZero ( )
```

Returns a copy of the data with zero points.

**Returns**

Data

**4.2.3.4 getDim()**

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

Returns the dimension of the dataset.

**Returns**

int

#### 4.2.3.5 getFeaturesNames()

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

Returns the features names.

##### Returns

std::vector<int>

#### 4.2.3.6 getIndex()

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

Returns the vector of indexes.

##### Returns

std::vector<int>

#### 4.2.3.7 getNumberNegativePoints()

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

Return the number of negative points.

##### Returns

int

#### 4.2.3.8 getNumberPositivePoints()

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

Return the number of positive points.

##### Returns

int

#### 4.2.3.9 getPoint()

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

Returns the point with the given index.

**Parameters**

<i>index</i>	Position of a point in the points array.
--------------	--

**Returns**

`std::vector<Points>`

**4.2.3.10 `getPoints()`**

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

Returns the vector of Points of the sample.

**Returns**

`std::vector<Points>`

**4.2.3.11 `getSize()`**

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

Returns the size of the dataset.

**Returns**

`int`

**4.2.3.12 `getStatistics()`**

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

Returns a class with the statistics info of the sample.

**Returns**

[Statistics](#)

**4.2.3.13 `insertPoint()` [1/2]**

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

Insert a point to the data from another sample.

## Parameters

<i>sample</i>	(???) Sample with the point to be added.
<i>id</i>	(???) Index of the point to be added.

## Returns

bool

**4.2.3.14** insertPoint() [2/2]

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

Insert a point to the end of vector points.

## Parameters

<i>p</i>	(???) <a href="#">Point</a> to be inserted.
----------	---

## Returns

bool

**4.2.3.15** isEmpty()

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

Returns if there's a dataset loaded.

## Returns

bool

**4.2.3.16** isNormalized()

```
bool Data::isNormalized ( )
```

Returns if the dataset is normalized.

## Returns

bool

#### 4.2.3.17 join()

```
void Data::join (
    Data data )
```

Merge one dataset with another.

##### Parameters

<i>data</i>	(???) Dataset to be joined.
-------------	-----------------------------

##### Returns

bool

#### 4.2.3.18 load()

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

Load a dataset from a file.

##### Parameters

<i>file</i>	(???) Path to dataset file.
-------------	-----------------------------

##### Returns

bool

#### 4.2.3.19 normalize() [1/2]

```
void Data::normalize (
    double p = 2 )
```

normalize Normalize the dataset using a Lp-norm.

##### Parameters

<i>p</i>	Norm to be utilized.
----------	----------------------



**4.2.3.20** `normalize()` [2/2]

```
static void Data::normalize (
    std::vector< double > & p,
    double q ) [static]
```

`normalize` Normalize a vector using a Lp-norm.

**Parameters**

<i>q</i>	Norm to be utilized.
<i>p</i>	Vector to be normalized.

**4.2.3.21** `removeFeatures()`

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

Remove several features from the sample.

**Parameters**

<i>feats</i>	(???) Names of the features to be removed (must be sorted).
--------------	---

**Returns**

boolean informing if all features were succesfully removed.

**4.2.3.22** `removePoint()`

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

Remove a point from the data.

**Parameters**

<i>pid</i>	(???) Index of the point to be removed.
------------	---

**Returns**

bool

#### 4.2.3.23 removePoints()

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

Remove several points from the sample.

##### Parameters

<i>ids</i>	(???) Ids of the points to be removed (must be sorted).
------------	---

##### Returns

booleans informing which points were removed succesfully.

#### 4.2.3.24 setClasses()

```
void Data::setClasses (
    std::string pos,
    std::string neg )
```

setClasses Set the classes of the dataset.

##### Parameters

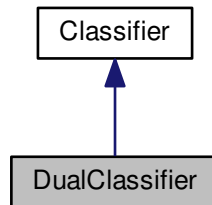
<i>pos</i>	Positive class.
<i>neg</i>	Negative class.

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

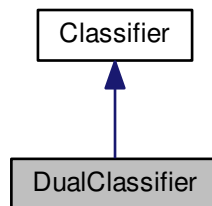
- includes/Data.hpp
- src/[Data.cpp](#)

## 4.3 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

## 4.4 FeatureSelection Class Reference

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

- includes/FeatureSelection.hpp

## 4.5 Gnuplot Class Reference

### Public Member Functions

- [Gnuplot](#) (const std::string &style="points")  
*set a style during construction*
- [Gnuplot](#) (const std::vector< double > &x, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")  
*plot a single std::vector at one go*
- [Gnuplot](#) (const std::vector< double > &x, const std::vector< double > &y, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")  
*plot pairs std::vector at one go*
- [Gnuplot](#) (const std::vector< double > &x, const std::vector< double > &y, const std::vector< double > &z, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y", const std::string &labelz="z")  
*plot triples std::vector at one go*
- [~Gnuplot](#) ()  
*destructor: needed to delete temporary files*
- [Gnuplot & cmd](#) (const std::string &cmdstr)  
*send a command to gnuplot*
- [Gnuplot & operator<<](#) (const std::string &cmdstr)  
*Sends a command to an active gnuplot session, identical to [cmd\(\)](#) send a command to gnuplot using the << operator.*
- [Gnuplot & showonscreen](#) ()  
*sets terminal type to terminal\_std*
- [Gnuplot & savetops](#) (const std::string &filename="gnuplot\_output")  
*saves a gnuplot session to a postscript file, filename without extension*
- [Gnuplot & set\\_style](#) (const std::string &stylestr="points")
- [Gnuplot & set\\_smooth](#) (const std::string &stylestr="csplines")
- [Gnuplot & unset\\_smooth](#) ()  
*unset smooth attention: smooth is not set by default*
- [Gnuplot & set\\_pointsize](#) (const double pointsize=1.0)  
*scales the size of the points used in plots*
- [Gnuplot & set\\_grid](#) ()  
*turns grid on/off*
- [Gnuplot & unset\\_grid](#) ()  
*grid is not set by default*
- [Gnuplot & set\\_multiplot](#) ()
- [Gnuplot & unset\\_multiplot](#) ()
- [Gnuplot & set\\_samples](#) (const int samples=100)  
*set sampling rate of functions, or for interpolating data*
- [Gnuplot & set\\_isosamples](#) (const int isolines=10)  
*set isoline density (grid) for plotting functions as surfaces (for 3d plots)*
- [Gnuplot & set\\_hidden3d](#) ()
- [Gnuplot & unset\\_hidden3d](#) ()
- [Gnuplot & set\\_contour](#) (const std::string &position="base")
- [Gnuplot & unset\\_contour](#) ()
- [Gnuplot & set\\_surface](#) ()
- [Gnuplot & unset\\_surface](#) ()
- [Gnuplot & set\\_legend](#) (const std::string &position="default")
- [Gnuplot & unset\\_legend](#) ()  
*Switches legend off attention: legend is set by default.*
- [Gnuplot & set\\_title](#) (const std::string &title="")

- sets and clears the title of a gnuplot session*
- [Gnuplot & unset\\_title](#) ()
  - Clears the title of a gnuplot session The title is not set by default.*
- [Gnuplot & set\\_ylabel](#) (const std::string &label="x")
  - set x axis label*
- [Gnuplot & set\\_xlabel](#) (const std::string &label="y")
  - set y axis label*
- [Gnuplot & set\\_zlabel](#) (const std::string &label="z")
  - set z axis label*
- [Gnuplot & set\\_xrange](#) (const double iFrom, const double iTo)
  - set axis - ranges*
- [Gnuplot & set\\_yrange](#) (const double iFrom, const double iTo)
  - set y-axis - ranges*
- [Gnuplot & set\\_zrange](#) (const double iFrom, const double iTo)
  - set z-axis - ranges*
- [Gnuplot & set\\_xautoscale](#) ()
- [Gnuplot & set\\_yautoscale](#) ()
- [Gnuplot & set\\_zautoscale](#) ()
- [Gnuplot & set\\_xlogscale](#) (const double base=10)
  - turns on/off log scaling for the specified xaxis (logscale is not set by default)*
- [Gnuplot & set\\_ylogscale](#) (const double base=10)
  - turns on/off log scaling for the specified yaxis (logscale is not set by default)*
- [Gnuplot & set\\_zlogscale](#) (const double base=10)
  - turns on/off log scaling for the specified zaxis (logscale is not set by default)*
- [Gnuplot & unset\\_xlogscale](#) ()
- [Gnuplot & unset\\_ylogscale](#) ()
- [Gnuplot & unset\\_zlogscale](#) ()
- [Gnuplot & set\\_cbrange](#) (const double iFrom, const double iTo)
  - set palette range (autoscale by default)*
- [Gnuplot & plotfile\\_x](#) (const std::string &filename, const unsigned int column=1, const std::string &title="")
- `template<typename X >`  
[Gnuplot & plot\\_x](#) (const X &x, const std::string &title="")
  - from std::vector*
- [Gnuplot & plotfile\\_xy](#) (const std::string &filename, const unsigned int column\_x=1, const unsigned int column\_y=2, const std::string &title="")
- `template<typename X , typename Y >`  
[Gnuplot & plot\\_xy](#) (const X &x, const Y &y, const std::string &title="")
  - from data*
- [Gnuplot & plotfile\\_xy\\_err](#) (const std::string &filename, const unsigned int column\_x=1, const unsigned int column\_y=2, const unsigned int column\_dy=3, const std::string &title="")
- `template<typename X , typename Y , typename E >`  
[Gnuplot & plot\\_xy\\_err](#) (const X &x, const Y &y, const E &dy, const std::string &title="")
  - from data*
- [Gnuplot & plotfile\\_xyz](#) (const std::string &filename, const unsigned int column\_x=1, const unsigned int column\_y=2, const unsigned int column\_z=3, const std::string &title="")
- `template<typename X , typename Y , typename Z >`  
[Gnuplot & plot\\_xyz](#) (const X &x, const Y &y, const Z &z, const std::string &title="")
  - from std::vector*
- [Gnuplot & plot\\_slope](#) (const double a, const double b, const std::string &title="")
  - plot an equation of the form:  $y = ax + b$ , you supply a and b*
- [Gnuplot & plot\\_equation](#) (const std::string &equation, const std::string &title="")
- [Gnuplot & plot\\_equation3d](#) (const std::string &equation, const std::string &title="")

- [Gnuplot](#) & [plot\\_image](#) (const unsigned char \*ucPicBuf, const unsigned int iWidth, const unsigned int iHeight, const std::string &title="")  
*plot image*
- [Gnuplot](#) & [replot](#) (void)  
*replot repeats the last plot or splot command. this can be useful for viewing a plot with different set options, or when generating the same plot for several devices (showonscreen, savetops)*
- [Gnuplot](#) & [reset\\_plot](#) ()  
*resets a gnuplot session (next plot will erase previous ones)*
- [Gnuplot](#) & [reset\\_all](#) ()  
*resets a gnuplot session and sets all variables to default*
- void [remove\\_tmpfiles](#) ()  
*deletes temporary files*
- bool [is\\_valid](#) ()  
*Is the gnuplot session valid ??*

## Static Public Member Functions

- static bool [set\\_GNUPlotPath](#) (const std::string &path)  
*optional function: set [Gnuplot](#) path manual attention: for windows: path with slash '/' not backslash '\'*
- static void [set\\_terminal\\_std](#) (const std::string &type)

## 4.5.1 Member Function Documentation

### 4.5.1.1 [is\\_valid\(\)](#)

```
bool Gnuplot::is_valid ( ) [inline]
```

Is the gnuplot session valid ??

#### Parameters

—	
---	--

#### Returns

true if valid, false if not

### 4.5.1.2 [operator<<\(\)](#)

```
Gnuplot& Gnuplot::operator<< (
    const std::string & cmdstr ) [inline]
```

Sends a command to an active gnuplot session, identical to [cmd\(\)](#) send a command to gnuplot using the << operator.

## Parameters

<i>cmdstr</i>	→ the command string
---------------	----------------------

## Returns

← a reference to the gnuplot object

## 4.5.1.3 plot\_equation()

```
Gnuplot & Gnuplot::plot_equation (
    const std::string & equation,
    const std::string & title = "" )
```

plot an equation supplied as a std::string y=f(x), write only the function f(x) not y= the independent variable has to be x binary operators: \*\* exponentiation, \* multiply, / divide, + add, - subtract, % modulo unary operators: - minus, ! factorial elementary functions: rand(x), abs(x), sgn(x), ceil(x), floor(x), int(x), imag(x), real(x), arg(x), sqrt(x), exp(x), log(x), log10(x), sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(y,x), sinh(x), cosh(x), tanh(x), asinh(x), acosh(x), atanh(x) special functions: erf(x), erfc(x), inverf(x), gamma(x), igamma(a,x), lgamma(x), ibeta(p,q,x), besj0(x), besj1(x), besy0(x), besy1(x), lambertw(x) statistical fuctions: norm(x), invnorm(x)

## 4.5.1.4 plot\_equation3d()

```
Gnuplot & Gnuplot::plot_equation3d (
    const std::string & equation,
    const std::string & title = "" )
```

plot an equation supplied as a std::string z=f(x,y), write only the function f(x,y) not z= the independent variables have to be x and y

## 4.5.1.5 plot\_image()

```
Gnuplot & Gnuplot::plot_image (
    const unsigned char * ucPicBuf,
    const unsigned int iWidth,
    const unsigned int iHeight,
    const std::string & title = "" )
```

plot image

- note that this function is not valid for versions of GNUPlot below 4.2

#### 4.5.1.6 plot\_x()

```
template<typename X >
Gnuplot & Gnuplot::plot_x (
    const X & x,
    const std::string & title = "" )
```

from std::vector

Plots a 2d graph from a list of doubles: x.

#### 4.5.1.7 plot\_xy()

```
template<typename X , typename Y >
Gnuplot & Gnuplot::plot_xy (
    const X & x,
    const Y & y,
    const std::string & title = "" )
```

from data

Plots a 2d graph from a list of doubles: x y.

#### 4.5.1.8 plot\_xy\_err()

```
template<typename X , typename Y , typename E >
Gnuplot & Gnuplot::plot_xy_err (
    const X & x,
    const Y & y,
    const E & dy,
    const std::string & title = "" )
```

from data

plot x,y pairs with dy errorbars

#### 4.5.1.9 plotfile\_x()

```
Gnuplot & Gnuplot::plotfile_x (
    const std::string & filename,
    const unsigned int column = 1,
    const std::string & title = "" )
```

plot a single std::vector: x from file

#### 4.5.1.10 plotfile\_xy()

```
Gnuplot & Gnuplot::plotfile_xy (
    const std::string & filename,
    const unsigned int column_x = 1,
    const unsigned int column_y = 2,
    const std::string & title = "" )
```

plot x,y pairs: x y from file



4.5.1.11 `plotfile_xy_err()`

```
Gnuplot & Gnuplot::plotfile_xy_err (
    const std::string & filename,
    const unsigned int column_x = 1,
    const unsigned int column_y = 2,
    const unsigned int column_dy = 3,
    const std::string & title = "" )
```

plot x,y pairs with dy errorbars: x y dy from file

4.5.1.12 `plotfile_xyz()`

```
Gnuplot & Gnuplot::plotfile_xyz (
    const std::string & filename,
    const unsigned int column_x = 1,
    const unsigned int column_y = 2,
    const unsigned int column_z = 3,
    const std::string & title = "" )
```

plot x,y,z triples: x y z from file

4.5.1.13 `replot()`

```
Gnuplot& Gnuplot::replot (
    void ) [inline]
```

replot repeats the last plot or splot command. this can be useful for viewing a plot with different set options, or when generating the same plot for several devices (showonscreen, savetops)

## Parameters

—	
---	--

## Returns

—

4.5.1.14 `set_contour()`

```
Gnuplot & Gnuplot::set_contour (
    const std::string & position = "base" )
```

enables/disables contour drawing for surfaces (for 3d plot) base, surface, both

4.5.1.15 `set_GNUPlotPath()`

```
bool Gnuplot::set_GNUPlotPath (
    const std::string & path ) [static]
```

optional function: set [Gnuplot](#) path manual attention: for windows: path with slash '/' not backslash '\'

**Parameters**

<i>path</i>	→ the gnuplot path
-------------	--------------------

**Returns**

true on success, false otherwise

**4.5.1.16 set\_hidden3d()**

```
Gnuplot& Gnuplot::set_hidden3d ( ) [inline]
```

enables/disables hidden line removal for surface plotting (for 3d plot)

**Parameters**

—	
---	--

**Returns**

← reference to the gnuplot object

**4.5.1.17 set\_legend()**

```
Gnuplot & Gnuplot::set_legend (
    const std::string & position = "default" )
```

switches legend on/off position: inside/outside, left/center/right, top/center/bottom, nobox/box

**4.5.1.18 set\_multiplot()**

```
Gnuplot& Gnuplot::set_multiplot ( ) [inline]
```

set the mulitplot mode

**Parameters**

—	
---	--

**Returns**

← reference to the gnuplot object

4.5.1.19 `set_smooth()`

```
Gnuplot & Gnuplot::set_smooth (
    const std::string & stylestr = "csplines" )
```

interpolation and approximation of data, arguments: csplines, bezier, acsplines (for data values > 0), sbezier, unique, frequency (works only with plot\_x, plot\_xy, plotfile\_x, plotfile\_xy (if smooth is set, set\_style has no effect on data plotting)

4.5.1.20 `set_style()`

```
Gnuplot & Gnuplot::set_style (
    const std::string & stylestr = "points" )
```

set line style (some of these styles require additional information): lines, points, linespoints, impulses, dots, steps, fsteps, histeps, boxes, histograms, filledcurves

4.5.1.21 `set_surface()`

```
Gnuplot & Gnuplot::set_surface ( ) [inline]
```

enables/disables the display of surfaces (for 3d plot)

## Parameters

—	
---	--

## Returns

← reference to the gnuplot object

4.5.1.22 `set_terminal_std()`

```
void Gnuplot::set_terminal_std (
    const std::string & type ) [static]
```

optional: set standart terminal, used by showonscreen defaults: Windows - win, Linux - x11, Mac - aqua

## Parameters

<i>type</i>	→ the terminal type
-------------	---------------------

## Returns

—

#### 4.5.1.23 `set_title()`

```
Gnuplot& Gnuplot::set_title (
    const std::string & title = "" ) [inline]
```

sets and clears the title of a gnuplot session

##### Parameters

<i>title</i>	→ the title of the plot [optional, default == ""]
--------------	---

##### Returns

← reference to the gnuplot object

#### 4.5.1.24 `set_xautoscale()`

```
Gnuplot& Gnuplot::set_xautoscale ( ) [inline]
```

autoscale axis (set by default) of xaxis

##### Parameters

—	
---	--

##### Returns

← reference to the gnuplot object

#### 4.5.1.25 `set_yautoscale()`

```
Gnuplot& Gnuplot::set_yautoscale ( ) [inline]
```

autoscale axis (set by default) of yaxis

##### Parameters

—	
---	--

##### Returns

← reference to the gnuplot object

#### 4.5.1.26 set\_zautoscale()

```
Gnuplot& Gnuplot::set_zautoscale ( ) [inline]
```

autoscale axis (set by default) of zaxis

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.27 unset\_contour()

```
Gnuplot& Gnuplot::unset_contour ( ) [inline]
```

contour is not set by default, it disables contour drawing for surfaces

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.28 unset\_hidden3d()

```
Gnuplot& Gnuplot::unset_hidden3d ( ) [inline]
```

hidden3d is not set by default

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.29 unset\_legend()

`Gnuplot& Gnuplot::unset_legend ( ) [inline]`

Switches legend off attention: legend is set by default.

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.30 unset\_multiplot()

`Gnuplot& Gnuplot::unset_multiplot ( ) [inline]`

unsets the mulitplot mode

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.31 unset\_smooth()

`Gnuplot& Gnuplot::unset_smooth ( ) [inline]`

unset smooth attention: smooth is not set by default

##### Parameters

—	
---	--

##### Returns

<— a reference to a gnuplot object

#### 4.5.1.32 unset\_surface()

`Gnuplot`& `Gnuplot::unset_surface ( )` [inline]

surface is set by default, it disables the display of surfaces (for 3d plot)

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.33 unset\_title()

`Gnuplot`& `Gnuplot::unset_title ( )` [inline]

Clears the title of a gnuplot session The title is not set by default.

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.34 unset\_xlogscale()

`Gnuplot`& `Gnuplot::unset_xlogscale ( )` [inline]

turns off log scaling for the x axis

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.35 unset\_ylogscale()

`Gnuplot& Gnuplot::unset_ylogscale ( ) [inline]`

turns off log scaling for the y axis

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

#### 4.5.1.36 unset\_zlogscale()

`Gnuplot& Gnuplot::unset_zlogscale ( ) [inline]`

turns off log scaling for the z axis

##### Parameters

—	
---	--

##### Returns

<— reference to the gnuplot object

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

- includes/gnuplot\_i.hpp
- src/gnuplot\_i.cpp

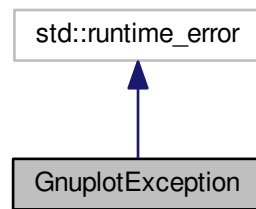
## 4.6 GnuplotException Class Reference

A C++ interface to gnuplot.

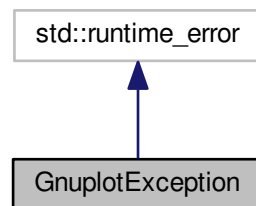
```
#include <gnuplot_i.hpp>
```



Inheritance diagram for GnuplotException:



Collaboration diagram for GnuplotException:



## Public Member Functions

- **GnuplotException** (const std::string &msg)

### 4.6.1 Detailed Description

A C++ interface to gnuplot.

The interface uses pipes and so won't run on a system that doesn't have POSIX pipe support Tested on Windows (MinGW and Visual C++) and Linux (GCC)

Version history: 0. C interface by N. Devillard (27/01/03)

1. C++ interface: direct translation from the C interface by Rajarshi Guha (07/03/03)
2. corrections for Win32 compatibility by V. Chyzhdzenka (20/05/03)
3. some member functions added, corrections for Win32 and Linux compatibility by M. Burgis (10/03/08)
4. Move function definition into gnuplot\_i.cpp by X. BROQUERE (25/10/11)

Requirements:

- gnuplot has to be installed (<http://www.gnuplot.info/download.html>)
- for Windows: set Path-Variable for [Gnuplot](#) path (e.g. C:/program files/gnuplot/bin) or set [Gnuplot](#) path with: [Gnuplot::set\\_GNUPlotPath\(const std::string &path\);](#)

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

- includes/gnuplot\_i.hpp

## 4.7 Kernel Class Reference

Class for the kernel computations.

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

### Public Member Functions

- double [norm](#) ([Data](#) data)  
*norm Computes norm in dual variables.*

#### 4.7.1 Detailed Description

Class for the kernel computations.

#### 4.7.2 Member Function Documentation

##### 4.7.2.1 norm()

```
double Kernel::norm (  
    Data data )
```

norm Computes norm in dual variables.

##### Parameters

<i>data</i>	Dataset to compute norm.
-------------	--------------------------

##### Returns

double

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

- includes/Kernel.hpp
- src/Kernel.cpp

## 4.8 Point Class Reference

Class of a [Point](#) of doubles in a space of n dimensions.

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

### Public Member Functions

- **Point** (int dim, int val=0)
- 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.*
- double **alpha** = 0.0
- int [id](#) = 0  
*[Point](#) identification.*

### Friends

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

#### 4.8.1 Detailed Description

Class of a [Point](#) of doubles in a space of n dimensions.

#### 4.8.2 Member Function Documentation

##### 4.8.2.1 dot()

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

Computes the dot product with a vector.

**Parameters**

$p$	(???)
-----	-------

**Returns**

double

**4.8.2.2 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**

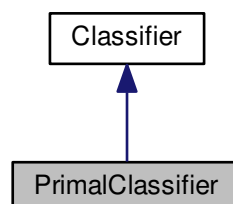
double

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

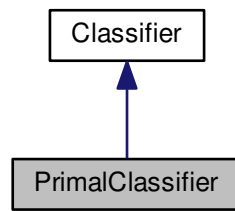
- includes/Point.hpp
- src/[Point.cpp](#)

## 4.9 PrimalClassifier Class Reference

Inheritance diagram for PrimalClassifier:



Collaboration diagram for PrimalClassifier:



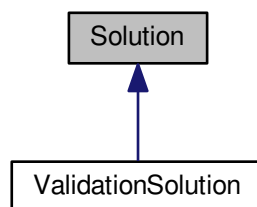
### Additional Inherited Members

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

- includes/PrimalClassifier.hpp

## 4.10 Solution Class Reference

Inheritance diagram for Solution:



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

- includes/Solution.hpp

## 4.11 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.*
- static double [getRadius](#) ([Data](#) data, int index, double q)  
*Returns radius of the ball that circ. the data.*
- static double [getDistCenters](#) ([Data](#) data, int index)  
*Returns distance of centers of the classes.*
- static double [getDistCentersWithoutFeats](#) ([Data](#) data, std::vector< int > feats, int index)  
*Returns distance of centers of the classes without given features.*

## Friends

- class **Data**

### 4.11.1 Detailed Description

Class with methods for statistical computations.

### 4.11.2 Member Function Documentation

#### 4.11.2.1 [getDistCenters\(\)](#)

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

Returns distance of centers of the classes.

#### Parameters

<i>data</i>	Dataset to compute the distance.
<i>index</i>	Feature to be ignored (-1 uses all features).

**Returns**

double

**4.11.2.2 getDistCentersWithoutFeats()**

```
double Statistics::getDistCentersWithoutFeats (
    Data data,
    std::vector< int > feats,
    int index ) [static]
```

Returns distance of centers of the classes without given features.

**Parameters**

<i>data</i>	Dataset to compute the distance.
<i>feats</i>	Features to be excluded from the computation.
<i>index</i>	Feature to be ignored (-1 uses all features).

**Returns**

double

**4.11.2.3 getFeatureMean()**

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

Computes the mean of a feature in the sample.

**Parameters**

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

**Returns**

double

**4.11.2.4 getFeatureStdev()**

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

Computes the standard deviation of a feature.

#### Parameters

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

#### Returns

double

#### 4.11.2.5 getRadius()

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

Returns radius of the ball that circ. the data.

#### Parameters

<i>data</i>	Dataset to compute the radius.
<i>index</i>	Feature to be ignored (-1 uses all features).
<i>q</i>	Lp-Norm to be used.

#### Returns

double

#### 4.11.2.6 mean()

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

Compute the mean (average) of a vector.

#### Parameters

<i>p</i>	(???) <a href="#">Point</a> to compute the mean.
----------	--

#### Returns

double



#### 4.11.2.7 stdev()

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

Compute the standard deviation of a vector.

##### Parameters

<i>p</i>	(???) <a href="#">Point</a> to compute stdev.
----------	---

##### Returns

double

#### 4.11.2.8 variance() [1/2]

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

Compute the variance of a vector.

##### Parameters

<i>p</i>	(???) Vector to compute the variance.
----------	---------------------------------------

##### Returns

double

#### 4.11.2.9 variance() [2/2]

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

Compute the variance of a sample.

##### Parameters

<i>data</i>	(???) Sample to compute the variance.
<i>index</i>	(???) 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

## 4.12 Validation Class Reference

Class of methods for the validation of ML algorithms.

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

**Public Member Functions**

- **Validation** ([Data](#) sample, [Classifier](#) \*classifier=NULL)
- void [partTrainTest](#) (int fold, uint seed)  
*Divide sample into train and test.*
- double **kFold** (int fold, int seed)
- void **validation** (int fold, int qtde)
- [Data](#) **getTestSample** ()
- [Data](#) **getTrainSample** ()

### 4.12.1 Detailed Description

Class of methods for the validation of ML algorithms.

### 4.12.2 Member Function Documentation

#### 4.12.2.1 [partTrainTest\(\)](#)

```
void Validation::partTrainTest (
    int fold,
    uint seed )
```

Divide sample into train and test.

**Parameters**

<i>fold</i>	Number of folds.
<i>seed</i>	Seed to feed the pseudo random number generator.

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

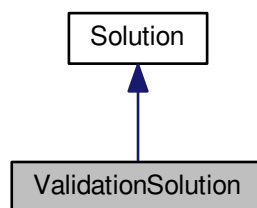
- includes/Validation.hpp
- src/Validation.cpp

## 4.13 ValidationSolution Class Reference

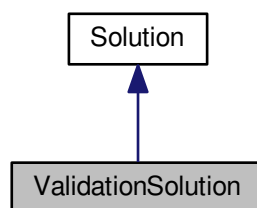
[Solution](#) for the validation of a ML method.

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

Inheritance diagram for ValidationSolution:



Collaboration diagram for ValidationSolution:



### 4.13.1 Detailed Description

[Solution](#) for the validation of a ML method.

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

- includes/ValidationSolution.hpp

## 4.14 Visualisation Class Reference

Class for visualize data using gnuplot.

```
#include <Visualisation.hpp>
```

### Public Member Functions

- **Visualisation** ([Data](#) \*sample)
- void [setSample](#) ([Data](#) sample)  
*Set sample to be visualized.*
- void [setTitle](#) (std::string title)  
*Set plot title.*
- void [setStyle](#) (std::string style)  
*Set plot style. (points, lines, etc.)*
- void [plot2D](#) (int x, int y)  
*Plot the selected features in 2D.*
- void [plot3D](#) (int x, int y, int z)  
*Plot the selected features in 3D.*

### 4.14.1 Detailed Description

Class for visualize data using gnuplot.

### 4.14.2 Member Function Documentation

#### 4.14.2.1 [plot2D\(\)](#)

```
void Visualisation::plot2D (
    int x,
    int y )
```

Plot the selected features in 2D.

#### Parameters

<i>x</i>	(???) Feature to be used in the x-axis.
<i>y</i>	(???) Feature to be used in the y-axis.

#### Returns

void

#### 4.14.2.2 plot3D()

```
void Visualisation::plot3D (
    int x,
    int y,
    int z )
```

Plot the selected features in 3D.

##### Parameters

<i>x</i>	(???) Feature to be used in the x-axis.
<i>y</i>	(???) Feature to be used in the y-axis.
<i>z</i>	(???) Feature to be used in the z-axis.

##### Returns

void

#### 4.14.2.3 setSample()

```
void Visualisation::setSample (
    Data sample )
```

Set sample to be visualized.

##### Parameters

<i>sample</i>	(???) <a href="#">Data</a> to set for visualization.
---------------	--

##### Returns

void

#### 4.14.2.4 setStyle()

```
void Visualisation::setStyle (
    std::string style )
```

Set plot style. (points, lines, etc.)

##### Parameters

<i>style</i>	(???) Style to be set.
--------------	------------------------

**Returns**

void

**4.14.2.5 setTitle()**

```
void Visualisation::setTitle (
    std::string title )
```

Set plot title.

**Parameters**

<i>title</i>	(???) Plot title.
--------------	-------------------

**Returns**

void

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

- includes/Visualisation.hpp
- src/Visualisation.cpp

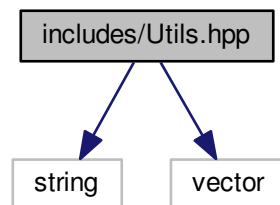
## Chapter 5

# File Documentation

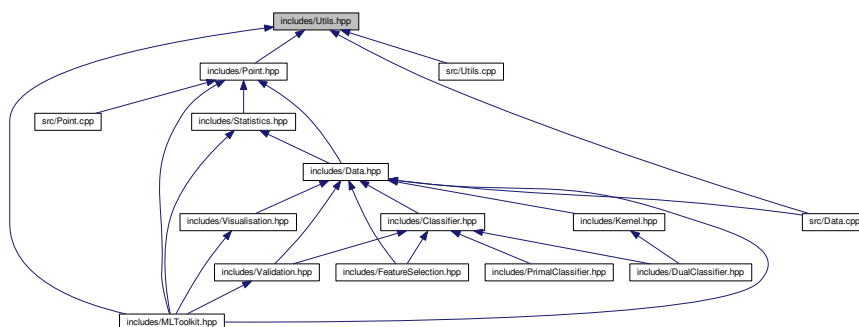
### 5.1 includes/Utils.hpp File Reference

```
#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`

## Typedefs

- `typedef std::vector< std::vector< double > > dMatrix`

## 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

<i>str</i>	String to be tested.
------------	----------------------



**Returns**

bool

**5.1.2.2 maxAbsElement()**

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

Returns the max absolute element.

**Parameters**

<i>x</i>	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**

<i>str</i>	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**

<i>str</i>	String to be converted.
------------	-------------------------

**Returns**

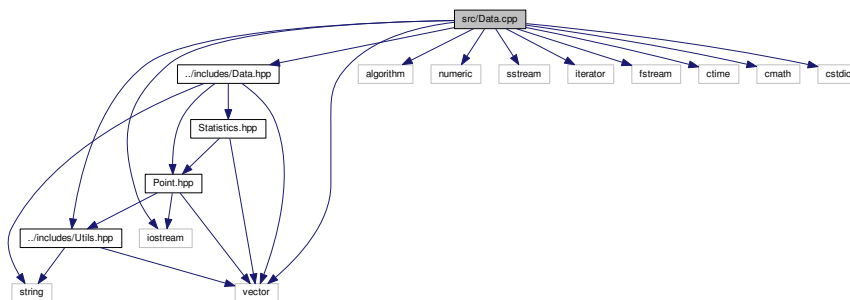
The integer resulted from the conversion.

**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:

**Functions**

- ostream & **operator**<< (ostream &output, const [Data](#) &data)

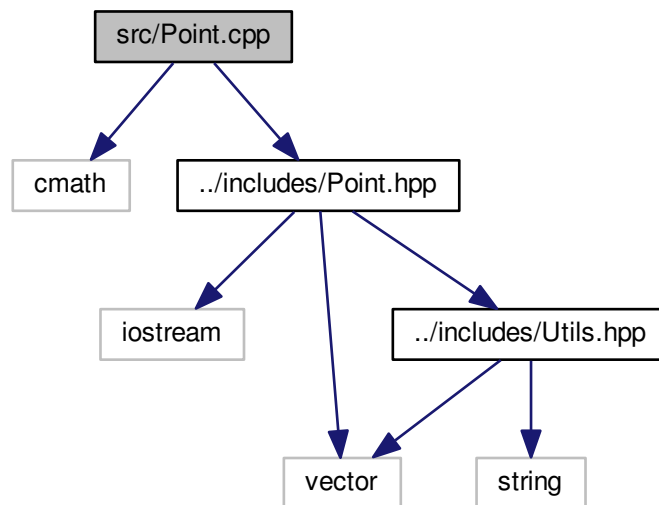
**5.2.1 Detailed Description**

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 dependency graph for Point.cpp:
```



### Functions

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

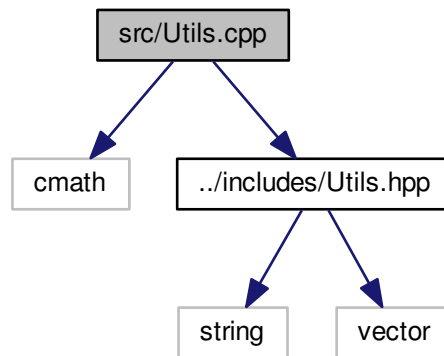
#### 5.3.1 Detailed Description

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

- changeXVector
  - Data, [10](#)
- Classifier, [7](#)
  - evaluate, [7](#)
  - train, [8](#)
- copy
  - Data, [12](#)
- copyZero
  - Data, [12](#)
- Data, [8](#)
  - changeXVector, [10](#)
  - copy, [12](#)
  - copyZero, [12](#)
  - Data, [10](#)
  - getDim, [12](#)
  - getFeaturesNames, [12](#)
  - getIndex, [13](#)
  - getNumberNegativePoints, [13](#)
  - getNumberPositivePoints, [13](#)
  - getPoint, [13](#)
  - getPoints, [14](#)
  - getSize, [14](#)
  - getStatistics, [14](#)
  - insertPoint, [14](#), [15](#)
  - isEmpty, [15](#)
  - isNormalized, [15](#)
  - join, [15](#)
  - load, [16](#)
  - normalize, [16](#)
  - removeFeatures, [17](#)
  - removePoint, [17](#)
  - removePoints, [17](#)
  - setClasses, [18](#)
- dot
  - Point, [35](#)
- DualClassifier, [19](#)
- evaluate
  - Classifier, [7](#)
- FeatureSelection, [19](#)
- getDim
  - Data, [12](#)
- getDistCenters
  - Statistics, [38](#)
- getDistCentersWithoutFeats
  - Statistics, [39](#)
- getFeatureMean
  - Statistics, [39](#)
- getFeatureStddev
  - Statistics, [39](#)
- getFeaturesNames
  - Data, [12](#)
- getIndex
  - Data, [13](#)
- getNumberNegativePoints
  - Data, [13](#)
- getNumberPositivePoints
  - Data, [13](#)
- getPoint
  - Data, [13](#)
- getPoints
  - Data, [14](#)
- getRadius
  - Statistics, [40](#)
- getSize
  - Data, [14](#)
- getStatistics
  - Data, [14](#)
- Gnuplot, [20](#)
  - is\_valid, [22](#)
  - operator<<, [22](#)
  - plot\_equation, [23](#)
  - plot\_equation3d, [23](#)
  - plot\_image, [23](#)
  - plot\_x, [23](#)
  - plot\_xy, [24](#)
  - plot\_xy\_err, [24](#)
  - plotfile\_x, [24](#)
  - plotfile\_xy, [24](#)
  - plotfile\_xy\_err, [24](#)
  - plotfile\_xyz, [25](#)
  - replot, [25](#)
  - set\_GNUPlotPath, [25](#)
  - set\_contour, [25](#)
  - set\_hidden3d, [26](#)
  - set\_legend, [26](#)
  - set\_multiplot, [26](#)
  - set\_smooth, [26](#)
  - set\_style, [27](#)
  - set\_surface, [27](#)
  - set\_terminal\_std, [27](#)
  - set\_title, [27](#)
  - set\_xautoscale, [28](#)
  - set\_yautoscale, [28](#)
  - set\_zautoscale, [28](#)
  - unset\_contour, [29](#)

- unset\_hidden3d, 29
- unset\_legend, 29
- unset\_multiplot, 30
- unset\_smooth, 30
- unset\_surface, 30
- unset\_title, 31
- unset\_xlogscale, 31
- unset\_ylogscale, 31
- unset\_zlogscale, 32
- GnuplotException, 32
- includes/Utils.hpp, 47
- insertPoint
  - Data, 14, 15
- is\_number
  - Utils.hpp, 48
- is\_valid
  - Gnuplot, 22
- isEmpty
  - Data, 15
- isNormalized
  - Data, 15
- join
  - Data, 15
- Kernel, 34
  - norm, 34
- load
  - Data, 16
- maxAbsElement
  - Utils.hpp, 49
- mean
  - Statistics, 40
- norm
  - Kernel, 34
  - Point, 36
- normalize
  - Data, 16
- operator<<
  - Gnuplot, 22
- partTrainTest
  - Validation, 42
- plot2D
  - Visualisation, 44
- plot3D
  - Visualisation, 44
- plot\_equation
  - Gnuplot, 23
- plot\_equation3d
  - Gnuplot, 23
- plot\_image
  - Gnuplot, 23
- plot\_x
  - Gnuplot, 23
- plot\_xy
  - Gnuplot, 24
- plot\_xy\_err
  - Gnuplot, 24
- plotfile\_x
  - Gnuplot, 24
- plotfile\_xy
  - Gnuplot, 24
- plotfile\_xy\_err
  - Gnuplot, 24
- plotfile\_xyz
  - Gnuplot, 25
- Point, 35
  - dot, 35
  - norm, 36
- PrimalClassifier, 36
- removeFeatures
  - Data, 17
- removePoint
  - Data, 17
- removePoints
  - Data, 17
- replot
  - Gnuplot, 25
- set\_GNUPlotPath
  - Gnuplot, 25
- set\_contour
  - Gnuplot, 25
- set\_hidden3d
  - Gnuplot, 26
- set\_legend
  - Gnuplot, 26
- set\_multiplot
  - Gnuplot, 26
- set\_smooth
  - Gnuplot, 26
- set\_style
  - Gnuplot, 27
- set\_surface
  - Gnuplot, 27
- set\_terminal\_std
  - Gnuplot, 27
- set\_title
  - Gnuplot, 27
- set\_xautoscale
  - Gnuplot, 28
- set\_yautoscale
  - Gnuplot, 28
- set\_zautoscale
  - Gnuplot, 28
- setClasses
  - Data, 18
- setSample
  - Visualisation, 45
- setStyle
  - Visualisation, 45
- setTitle

- Visualisation, [46](#)
- Solution, [37](#)
- src/Data.cpp, [50](#)
- src/Point.cpp, [51](#)
- src/Utils.cpp, [51](#)
- Statistics, [37](#)
  - getDistCenters, [38](#)
  - getDistCentersWithoutFeats, [39](#)
  - getFeatureMean, [39](#)
  - getFeatureStdev, [39](#)
  - getRadius, [40](#)
  - mean, [40](#)
  - stdev, [41](#)
  - variance, [41](#)
- stdev
  - Statistics, [41](#)
- stodn
  - Utils.hpp, [49](#)
- stoin
  - Utils.hpp, [49](#)
- train
  - Classifier, [8](#)
- unset\_contour
  - Gnuplot, [29](#)
- unset\_hidden3d
  - Gnuplot, [29](#)
- unset\_legend
  - Gnuplot, [29](#)
- unset\_multiplot
  - Gnuplot, [30](#)
- unset\_smooth
  - Gnuplot, [30](#)
- unset\_surface
  - Gnuplot, [30](#)
- unset\_title
  - Gnuplot, [31](#)
- unset\_xlogscale
  - Gnuplot, [31](#)
- unset\_ylogscale
  - Gnuplot, [31](#)
- unset\_zlogscale
  - Gnuplot, [32](#)
- Utils.hpp
  - is\_number, [48](#)
  - maxAbsElement, [49](#)
  - stodn, [49](#)
  - stoin, [49](#)
- Validation, [42](#)
  - partTrainTest, [42](#)
- ValidationSolution, [43](#)
- variance
  - Statistics, [41](#)
- Visualisation, [44](#)
  - plot2D, [44](#)
  - plot3D, [44](#)
  - setSample, [45](#)
  - setStyle, [45](#)
  - setTitle, [46](#)