

UM

Generated by Doxygen 1.8.18

1 Namespace Index	1
1.1 Packages	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 Build Namespace Reference	7
4.1.1 Function Documentation	7
4.1.1.1 build_tree()	7
4.1.1.2 count_all()	8
4.2 Data Namespace Reference	8
4.2.1 Function Documentation	8
4.2.1.1 read_csv_file()	8
4.2.1.2 read_data_file()	9
4.3 Data_matching Namespace Reference	9
4.3.1 Function Documentation	9
4.3.1.1 confusion_matrix()	9
4.3.1.2 data_matching()	10
4.3.1.3 find()	10
4.3.1.4 for_basic_tree()	11
4.3.1.5 for_tree_incremental_learning()	11
4.4 Incremental_learning Namespace Reference	11
4.4.1 Function Documentation	12
4.4.1.1 find_tree()	12
4.4.1.2 incremental_learning()	12
4.5 Mesure Namespace Reference	12
4.5.1 Function Documentation	13
4.5.1.1 count()	13
4.5.1.2 gain()	13
4.5.1.3 giny()	14
4.6 Print_tree Namespace Reference	14
4.6.1 Function Documentation	14
4.6.1.1 print_basic_tree()	14
4.6.1.2 print_incremental_tree()	15
4.6.1.3 print_tree()	15
4.7 Split Namespace Reference	15
4.7.1 Function Documentation	15
4.7.1.1 check_split()	16
4.7.1.2 make_split()	16

4.8 Test Namespace Reference	16
4.8.1 Function Documentation	17
4.8.1.1 data_matching_for_basic_tree()	17
4.8.1.2 data_matching_for_tree_incremental_learning()	17
4.8.1.3 read_bank()	19
4.8.1.4 test()	19
4.8.1.5 write_dict()	19
4.8.1.6 write_file()	20
4.8.1.7 write_time()	20
4.8.2 Variable Documentation	20
4.8.2.1 agaricus_incremental	20
4.8.2.2 bank	20
4.8.2.3 iris	20
5 Class Documentation	21
5.1 Build.Quantity Class Reference	21
5.1.1 Detailed Description	21
5.1.2 Constructor & Destructor Documentation	21
5.1.2.1 __init__()	21
5.1.3 Member Data Documentation	22
5.1.3.1 quantity	22
5.2 Build.Subtree_Values Class Reference	22
5.2.1 Detailed Description	22
5.2.2 Constructor & Destructor Documentation	22
5.2.2.1 __init__()	23
5.2.3 Member Data Documentation	23
5.2.3.1 false_data	23
5.2.3.2 gain	23
5.2.3.3 left_next	23
5.2.3.4 question	23
5.2.3.5 right_next	24
5.2.3.6 true_data	24
6 File Documentation	25
6.1 Build.py File Reference	25
6.2 Data.py File Reference	25
6.3 Data_matching.py File Reference	26
6.4 Incremental_learning.py File Reference	26
6.5 Mesure.py File Reference	26
6.6 Print_tree.py File Reference	27
6.7 Split.py File Reference	27
6.8 Test.py File Reference	27

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Build	7
Data	8
Data_matching	9
Incremental_learning	11
Measure	12
Print_tree	14
Split	15
Test	16

Chapter 2

Class Index

2.1 Class List

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

Build.Quantity	
Class with a dictionary in which the number of elements	21
Build.Subtree_Values	
Class with elemnet to save in all knots	22

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Build.py	25
Data.py	25
Data_matching.py	26
Incremental_learning.py	26
Mesure.py	26
Print_tree.py	27
Split.py	27
Test.py	27

Chapter 4

Namespace Documentation

4.1 Build Namespace Reference

Classes

- class [Quantity](#)
class with a dictionary in which the number of elements
- class [Subtree_Values](#)
class with elemnet to save in all knots

Functions

- def [count_all](#) (training_data)
function to count all elements
- def [build_tree](#) (training_data)
function to build tree

4.1.1 Function Documentation

4.1.1.1 build_tree()

```
def Build.build_tree (  
    training_data )
```

function to build tree

Parameters

<i>training_elements</i>	list of training elements
--------------------------	---------------------------

Returns

object [Subtree_Values](#) for knot
 object [Quantity](#) for knot

4.1.1.2 count_all()

```
def Build.count_all (
    training_data )
```

function to count all elements

Parameters

<i>training_elements</i>	list in which items will be counted
--------------------------	-------------------------------------

Returns

quantity of elements all elements

4.2 Data Namespace Reference**Functions**

- def [read_csv_file](#) (name)
function to read csv file
- def [read_data_file](#) (name)
function to read data file

4.2.1 Function Documentation**4.2.1.1 read_csv_file()**

```
def Data.read_csv_file (
    name )
```

function to read csv file

Parameters

<i>name</i>	name of data file
-------------	-------------------

Returns

data readed data

4.2.1.2 read_data_file()

```
def Data.read_data_file (
    name )
```

function to read data file

Parameters

<i>name</i>	name of data file
-------------	-------------------

Returns

data readed data

4.3 Data_matching Namespace Reference

Functions

- def [for_basic_tree](#) (quantity, data)
function to do test for basic tree
- def [for_tree_incremental_learning](#) (quantity_basic_tree, quantity, data)
function to do test for tree with one incremental learning
- def [confusion_matrix](#) (data_test, tree)
function for creating confusion matrix
- def [data_matching](#) (tree, data_test)
function for calculating test data matching
- def [find](#) (tree, dt)
find the list for test data

4.3.1 Function Documentation**4.3.1.1 confusion_matrix()**

```
def Data_matching.confusion_matrix (
    data_test,
    tree )
```

function for creating confusion matrix

Parameters

<i>tree</i>	
<i>data</i>	test

Returns

matrix confusion_matrix

4.3.1.2 data_matching()

```
def Data_matching.data_matching (
    tree,
    data_test )
```

function for calculating test data matching

Parameters

<i>tree</i>	
<i>data</i>	test

Returns

match

4.3.1.3 find()

```
def Data_matching.find (
    tree,
    dt )
```

find the list for test data

Parameters

<i>trees</i>	
<i>dt</i>	element for which we are looking for a place

Returns

tree[1].quantity leaf value

4.3.1.4 for_basic_tree()

```
def Data_matching.for_basic_tree (
    quantity,
    data )
```

function to do test for basic tree

Parameters

<i>quantity</i>	quantity of data used to tarin dree
-----------------	-------------------------------------

Returns

match matching basic tree
matrix confusion matrix

4.3.1.5 for_tree_incremental_learning()

```
def Data_matching.for_tree_incremental_learning (
    quantity_basic_tree,
    quantity,
    data )
```

function to do test for tree with one incremental learning

Parameters

<i>quantity_basic_tree</i>	quantity of data used to tarin basic tree
<i>quantity</i>	quantity of data used to incremental learning

Returns

match matching for incremental learning
matrix confusion matrix

4.4 Incremental_learning Namespace Reference

Functions

- def [find_tree](#) (tree, data)
function to find the same tree in old tree
- def [incremental_learning](#) (data, tree)
function for incremental learning

4.4.1 Function Documentation

4.4.1.1 find_tree()

```
def Incremental_learning.find_tree (
    tree,
    data )
```

function to find the same tree in old tree

Parameters

<i>tree</i>	- old tree
<i>data</i>	current data

Returns

tree tree from old tree which can be used

4.4.1.2 incremental_learning()

```
def Incremental_learning.incremental_learning (
    data,
    tree )
```

function for incremental learning

Parameters

<i>tree</i>	- old tree
<i>data</i>	current data

Returns

object Subtree_Values for knot

object Quantity for knot

4.5 Mesure Namespace Reference

Functions

- def [count](#) (training_elements)
function to count elements in all attributes

- def `giny` (training_data)
function to calculate the gini coefficient
- def `gain` (false, true, current)
function to calculate the information gain

4.5.1 Function Documentation

4.5.1.1 `count()`

```
def Measure.count (
    training_elements )
```

function to count elements in all attributes

Parameters

<i>training_elements</i>	list in which items will be counted
--------------------------	-------------------------------------

Returns

count_data dictionary with quantity of elements in all category

4.5.1.2 `gain()`

```
def Measure.gain (
    false,
    true,
    current )
```

function to calculate the information gain

Parameters

<i>false</i>	list of false elements in which the information gain will be counted
<i>true</i>	list of true elements in which the information gain will be counted
<i>current</i>	all list of true elements in which the information gain will be counted

Returns

info_gain information gain for current split

4.5.1.3 giny()

```
def Mesure.giny (
    training_data )
```

function to calculate the gini coefficient

Parameters

<i>training_elements</i>	list in which items will be counted
--------------------------	-------------------------------------

Returns

1 - giny_tmp gini coefficient

4.6 Print_tree Namespace Reference

Functions

- def [print_tree](#) (element, space="")
function to print tree
- def [print_basic_tree](#) (start, quantity, data)
function to build and print tree
- def [print_incremental_tree](#) (start, quantity, tree, data)
function to incremental learning and print tree

4.6.1 Function Documentation

4.6.1.1 print_basic_tree()

```
def Print_tree.print_basic_tree (
    start,
    quantity,
    data )
```

function to build and print tree

Parameters

<i>start</i>	- the data number in the csv file used to train the tree
<i>quantity</i>	- quantity of data used to train the tree
<i>data</i>	data for incremental learning

4.6.1.2 `print_incremental_tree()`

```
def Print_tree.print_incremental_tree (
    start,
    quantity,
    tree,
    data )
```

function to incremental learning and print tree

Parameters

<i>start</i>	- the data number in the csv file used to train the tree
<i>quantity</i>	- quantity of data used to train the tree
<i>tree</i>	- old tree
<i>data</i>	data for incremental learning

4.6.1.3 `print_tree()`

```
def Print_tree.print_tree (
    element,
    space = "" )
```

function to print tree

Parameters

<i>element</i>	tree
<i>space</i>	

4.7 Split Namespace Reference

Functions

- def [make_split](#) (training_data)
function to find the best split
- def [check_split](#) (training_data, question_split)
function to do split for only one question

4.7.1 Function Documentation

4.7.1.1 check_split()

```
def Split.check_split (
    training_data,
    question_split )
```

function to do split for only one question

Parameters

<i>training_elements</i>	list in which items will be split
--------------------------	-----------------------------------

Returns

best_gain_value the best find gain
 best_question_split the best find question to split
 best_true_data the best find list with true data
 best_false_data the best find list with false data

4.7.1.2 make_split()

```
def Split.make_split (
    training_data )
```

function to find the best split

Parameters

<i>training_elements</i>	list in which items will be split
--------------------------	-----------------------------------

Returns

best_gain_value the best find gain
 best_question_split the best find question to split
 best_true_data the best find list with true data
 best_false_data the best find list with false data

4.8 Test Namespace Reference

Functions

- def [read_bank](#) ()
function to read and optimization bank data
- def [write_file](#) (name, match)
function to write match

- def `write_time` (name, time)
function to write time
- def `write_dict` (matrix, name)
function to write confusion matrix
- def `data_matching_for_basic_tree` (quantity, name, data)
function for sending tests
- def `data_matching_for_tree_incremental_learning` (quantity_basic_tree, quantity, quantity_of_all, name, data)
function for sending tests
- def `test` ()
function to do all tests

Variables

- def `bank` = `read_bank`()
- `agaricus_incremental` = `Data.read_data_file`('agaricus-lepiota.data')
- `iris` = `Data.read_data_file`('iris.data')

4.8.1 Function Documentation

4.8.1.1 `data_matching_for_basic_tree()`

```
def Test.data_matching_for_basic_tree (
    quantity,
    name,
    data )
```

function for sending tests

Parameters

<code>quantity</code>	quantity of data used to train tree
-----------------------	-------------------------------------

Returns

match matching basic tree

4.8.1.2 `data_matching_for_tree_incremental_learning()`

```
def Test.data_matching_for_tree_incremental_learning (
    quantity_basic_tree,
    quantity,
    quantity_of_all,
```

```
name,  
data )
```

function for sending tests

Parameters

<i>quantity_basic_tree</i>	quantity of data used to train basic tree
<i>quantity</i>	quantity of data used to incremental learning

Returns

match matching for incremental learning

4.8.1.3 read_bank()

```
def Test.read_bank ( )
```

function to read and optimization bank data

Returns

data

4.8.1.4 test()

```
def Test.test ( )
```

function to do all tests

4.8.1.5 write_dict()

```
def Test.write_dict (
    matrix,
    name )
```

function to write confusion matrix

Parameters

<i>matrix</i>	confusion matrix
<i>name</i>	name of data

4.8.1.6 write_file()

```
def Test.write_file (
    name,
    match )
```

function to write match

Parameters

<i>name</i>	name of data
<i>match</i>	

4.8.1.7 write_time()

```
def Test.write_time (
    name,
    time )
```

function to write time

Parameters

<i>name</i>	name of data
<i>time</i>	

4.8.2 Variable Documentation

4.8.2.1 agaricus_incremental

```
Test.agaricus_incremental = Data.read\_data\_file('agaricus-lepiota.data')
```

4.8.2.2 bank

```
def Test.bank = read\_bank()
```

4.8.2.3 iris

```
Test.iris = Data.read\_data\_file('iris.data')
```

Chapter 5

Class Documentation

5.1 Build.Quantity Class Reference

class with a dictionary in which the number of elements

Public Member Functions

- `def __init__ (self, data)`
save information about number of elements

Public Attributes

- `quantity`
number of elements

5.1.1 Detailed Description

class with a dictionary in which the number of elements

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `__init__()`

```
def Build.Quantity.__init__ (  
    self,  
    data )
```

save information about number of elements

5.1.3 Member Data Documentation

5.1.3.1 quantity

`Build.Quantity.quantity`

number of elements

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

- [Build.py](#)

5.2 Build.Subtree_Values Class Reference

class with elemnet to save in all knots

Public Member Functions

- `def __init__` (self, [question](#), [right_next](#), [left_next](#), [gain](#), [true_data](#), [false_data](#))
save information about knots

Public Attributes

- [question](#)
question used to divide data
- [right_next](#)
next right knots
- [left_next](#)
next left knots
- [gain](#)
gain of information obtained
- [true_data](#)
list with true data - that met the query
- [false_data](#)
list with false data - which did not match the query

5.2.1 Detailed Description

class with elemnet to save in all knots

5.2.2 Constructor & Destructor Documentation

5.2.2.1 `__init__()`

```
def Build.Subtree_Values.__init__ (
    self,
    question,
    right_next,
    left_next,
    gain,
    true_data,
    false_data )
```

save information about knots

5.2.3 Member Data Documentation

5.2.3.1 `false_data`

`Build.Subtree_Values.false_data`

list with false data - which did not match the query

5.2.3.2 `gain`

`Build.Subtree_Values.gain`

gain of information obtained

5.2.3.3 `left_next`

`Build.Subtree_Values.left_next`

next left knots

5.2.3.4 `question`

`Build.Subtree_Values.question`

question used to divide data

5.2.3.5 right_next

`Build.Subtree_Values.right_next`

next right knots

5.2.3.6 true_data

`Build.Subtree_Values.true_data`

list with true data - that met the query

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

- [Build.py](#)

Chapter 6

File Documentation

6.1 Build.py File Reference

Classes

- class [Build.Subtree_Values](#)
class with elemnet to save in all knots
- class [Build.Quantity](#)
class with a dictionary in which the number of elements

Namespaces

- [Build](#)

Functions

- def [Build.count_all](#) (training_data)
function to count all elements
- def [Build.build_tree](#) (training_data)
function to build tree

6.2 Data.py File Reference

Namespaces

- [Data](#)

Functions

- def [Data.read_csv_file](#) (name)
function to read csv file
- def [Data.read_data_file](#) (name)
function to read data file

6.3 Data_matching.py File Reference

Namespaces

- [Data_matching](#)

Functions

- def [Data_matching.for_basic_tree](#) (quantity, data)
function to do test for basic tree
- def [Data_matching.for_tree_incremental_learning](#) (quantity_basic_tree, quantity, data)
function to do test for tree with one incremental learning
- def [Data_matching.confusion_matrix](#) (data_test, tree)
function for creating confusion matrix
- def [Data_matching.data_matching](#) (tree, data_test)
function for calculating test data matching
- def [Data_matching.find](#) (tree, dt)
find the list for test data

6.4 Incremental_learning.py File Reference

Namespaces

- [Incremental_learning](#)

Functions

- def [Incremental_learning.find_tree](#) (tree, data)
function to find the same tree in old tree
- def [Incremental_learning.incremental_learning](#) (data, tree)
function for incremental learning

6.5 Measure.py File Reference

Namespaces

- [Measure](#)

Functions

- def [Measure.count](#) (training_elements)
function to count elements in all attributes
- def [Measure.giny](#) (training_data)
function to calculate the gini coefficient
- def [Measure.gain](#) (false, true, current)
function to calculate the information gain

6.6 Print_tree.py File Reference

Namespaces

- [Print_tree](#)

Functions

- def [Print_tree.print_tree](#) (element, space="")
function to print tree
- def [Print_tree.print_basic_tree](#) (start, quantity, data)
function to build and print tree
- def [Print_tree.print_incremental_tree](#) (start, quantity, tree, data)
function to incremental learning and print tree

6.7 Split.py File Reference

Namespaces

- [Split](#)

Functions

- def [Split.make_split](#) (training_data)
function to find the best split
- def [Split.check_split](#) (training_data, question_split)
function to do split for only one question

6.8 Test.py File Reference

Namespaces

- [Test](#)

Functions

- def [Test.read_bank](#) ()
function to read and optimization bank data
- def [Test.write_file](#) (name, match)
function to write match
- def [Test.write_time](#) (name, time)
function to write time
- def [Test.write_dict](#) (matrix, name)
function to write confusion matrix
- def [Test.data_matching_for_basic_tree](#) (quantity, name, data)
function for sending tests
- def [Test.data_matching_for_tree_incremental_learning](#) (quantity_basic_tree, quantity, quantity_of_all, name, data)
function for sending tests
- def [Test.test](#) ()
function to do all tests

Variables

- def `Test.bank` = `read_bank()`
- `Test.agaricus_incremental` = `Data.read_data_file('agaricus-lepiota.data')`
- `Test.iris` = `Data.read_data_file('iris.data')`

Index

- `__init__`
 - `Build.Quantity`, [21](#)
 - `Build.Subtree_Values`, [22](#)
- `agaricus_incremental`
 - `Test`, [20](#)
- `bank`
 - `Test`, [20](#)
- `Build`, [7](#)
 - `build_tree`, [7](#)
 - `count_all`, [8](#)
- `Build.py`, [25](#)
- `Build.Quantity`, [21](#)
 - `__init__`, [21](#)
 - `quantity`, [22](#)
- `Build.Subtree_Values`, [22](#)
 - `__init__`, [22](#)
 - `false_data`, [23](#)
 - `gain`, [23](#)
 - `left_next`, [23](#)
 - `question`, [23](#)
 - `right_next`, [23](#)
 - `true_data`, [24](#)
- `build_tree`
 - `Build`, [7](#)
- `check_split`
 - `Split`, [15](#)
- `confusion_matrix`
 - `Data_matching`, [9](#)
- `count`
 - `Mesure`, [13](#)
- `count_all`
 - `Build`, [8](#)
- `Data`, [8](#)
 - `read_csv_file`, [8](#)
 - `read_data_file`, [9](#)
- `Data.py`, [25](#)
- `Data_matching`, [9](#)
 - `confusion_matrix`, [9](#)
 - `data_matching`, [10](#)
 - `find`, [10](#)
 - `for_basic_tree`, [10](#)
 - `for_tree_incremental_learning`, [11](#)
- `data_matching`
 - `Data_matching`, [10](#)
- `Data_matching.py`, [26](#)
- `data_matching_for_basic_tree`
 - `Test`, [17](#)
- `data_matching_for_tree_incremental_learning`
 - `Test`, [17](#)
- `false_data`
 - `Build.Subtree_Values`, [23](#)
- `find`
 - `Data_matching`, [10](#)
- `find_tree`
 - `Incremental_learning`, [12](#)
- `for_basic_tree`
 - `Data_matching`, [10](#)
- `for_tree_incremental_learning`
 - `Data_matching`, [11](#)
- `gain`
 - `Build.Subtree_Values`, [23](#)
 - `Mesure`, [13](#)
- `giny`
 - `Mesure`, [13](#)
- `Incremental_learning`, [11](#)
 - `find_tree`, [12](#)
 - `incremental_learning`, [12](#)
- `incremental_learning`
 - `Incremental_learning`, [12](#)
- `Incremental_learning.py`, [26](#)
- `iris`
 - `Test`, [20](#)
- `left_next`
 - `Build.Subtree_Values`, [23](#)
- `make_split`
 - `Split`, [16](#)
- `Mesure`, [12](#)
 - `count`, [13](#)
 - `gain`, [13](#)
 - `giny`, [13](#)
- `Mesure.py`, [26](#)
- `print_basic_tree`
 - `Print_tree`, [14](#)
- `print_incremental_tree`
 - `Print_tree`, [14](#)
- `Print_tree`, [14](#)
 - `print_basic_tree`, [14](#)
 - `print_incremental_tree`, [14](#)
 - `print_tree`, [15](#)
- `print_tree`
 - `Print_tree`, [15](#)

Print_tree.py, [27](#)

quantity

Build.Quantity, [22](#)

question

Build.Subtree_Values, [23](#)

read_bank

Test, [19](#)

read_csv_file

Data, [8](#)

read_data_file

Data, [9](#)

right_next

Build.Subtree_Values, [23](#)

Split, [15](#)

check_split, [15](#)

make_split, [16](#)

Split.py, [27](#)

Test, [16](#)

agaricus_incremental, [20](#)

bank, [20](#)

data_matching_for_basic_tree, [17](#)

data_matching_for_tree_incremental_learning, [17](#)

iris, [20](#)

read_bank, [19](#)

test, [19](#)

write_dict, [19](#)

write_file, [19](#)

write_time, [20](#)

test

Test, [19](#)

Test.py, [27](#)

true_data

Build.Subtree_Values, [24](#)

write_dict

Test, [19](#)

write_file

Test, [19](#)

write_time

Test, [20](#)