# preprocessor Documentation

*Release unknown*

**Harvey Bastidas**

**Jun 04, 2020**

# Contents

Implements modular components for dataset preprocessing: a data-trimmer, a standardizer, a feature selector and a sliding window data generator.

CHAPTER 1

# Contents

## 1.1 License



## 1.2 Contributors

- Harvey Bastidas <harveybc@ingeni-us.com>

## 1.3 Changelog

### 1.3.1 Version 0.1

- Feature A added

- FIX: nasty bug #1729 fixed
- add your changes here!

## 1.4 preprocessor

### 1.4.1 preprocessor package

#### 1.4.1.1 Subpackages

#### preprocessor.data_trimmer package

#### Submodules

#### preprocessor.data_trimmer.data_trimmer module

This File contains the DataTrimmer class. To run this script uncomment or add the following lines in the [options.entry_points] section in setup.cfg:

**console_scripts =** data-trimmer = data_trimmer.__main__:main

Then run *python setup.py install* which will install the command *data-trimmer* inside your current environment.

**class** preprocessor.data_trimmer.data_trimmer.**DataTrimmer**(*conf*)
Bases: *preprocessor.preprocessor.Preprocessor*

The Data Trimmer preprocessor class

**core**()

**Core preprocessor task after starting the instance with the main method.** Decide from the arguments, what trimming method to call.

Args: args (obj): command line parameters as objects

**load_from_config**()

**parse_args**(*args*)
Parse command line parameters

**Parameters** **args** (*[str]*) – command line parameters as list of strings

**Returns** command line parameters namespace

**Return type** argparse.Namespace

**store**()
Save preprocessed data and the configuration of the preprocessor.

**trim_auto**()
Trims all the constant columns and trims all rows with consecutive zeroes from start and end of the input dataset

Returns: rows_t, cols_t (int,int): number of rows and columns trimmed

**trim_columns**()
Trims all the constant columns from the input dataset

**Returns** number of rows and columns trimmed

**Return type** rows_t, cols_t (int,int)

**trim_fixed_rows**(*from_start*, *from_end*)
Trims a configurable number of rows from the start or end of the input dataset

**Parameters**

- **from_start** (*int*) – number of rows to remove from start (ignored if auto_trim)
- **from_end** (*int*) – number of rows to remove from end (ignored if auto_trim)

**Returns** number of rows and columns trimmed

**Return type** rows_t, cols_t (int,int)

preprocessor.data_trimmer.data_trimmer.**run**(*args*)
Entry point for console_scripts

### Module contents

### preprocessor.feature_selector package

### Submodules

### preprocessor.feature_selector.feature_selector module

This File contains the FeatureSelector class. To run this script uncomment or add the following lines in the [options.entry_points] section in setup.cfg:

**console_scripts =** feature_selector = feature_selector.__main__:main

Then run *python setup.py install* which will install the command *feature_selector* inside your current environment.

**class** preprocessor.feature_selector.feature_selector.**FeatureSelector**(*conf*)
Bases: *preprocessor.preprocessor.Preprocessor*

The FeatureSelector preprocessor class

**core**()

**Core preprocessor task after starting the instance with the main method.** Decide from the arguments, what method to call.

Args: args (obj): command line parameters as objects

**feature_selection**()
Process the dataset.

**load_from_config**()
Process the dataset from a config file.

**parse_args**(*args*)
Parse command line parameters

**Parameters** **args** (*[str]*) – command line parameters as list of strings

**Returns** command line parameters namespace

**Return type** argparse.Namespace

**store**()
Save preprocessed data and the configuration of the preprocessor.

preprocessor.feature_selector.feature_selector.**run**(*args*)
Entry point for console_scripts

preprocessor.feature_selector.feature_selector.**score_func_classification**(*X*, *Y*)
Used to score the features for feature selection, for regression. To be used in the fFeatureSeclector.feature_selection() method.

preprocessor.feature_selector.feature_selector.**score_func_regression**(*X*, *Y*)
Used to score the features for feature selection, for regression. To be used in the fFeatureSeclector.feature_selection() method.

### Module contents

### preprocessor.sliding_window package

### Submodules

### preprocessor.sliding_window.sliding_window module

This File contains the SlidingWindow class. To run this script uncomment or add the following lines in the [options.entry_points] section in setup.cfg:

**console_scripts =** sliding_window = sliding_window.__main__:main

Then run *python setup.py install* which will install the command *sliding_window* inside your current environment.

**class** preprocessor.sliding_window.sliding_window.**SlidingWindow**(*conf*)
Bases: *preprocessor.preprocessor.Preprocessor*

The SlidingWindow preprocessor class

**core**()

**Core preprocessor task after starting the instance with the main method.** Decide from the arguments, what method to call.

Args: args (obj): command line parameters as objects

**parse_args**(*args*)
Parse command line parameters additional to the preprocessor class ones

**Parameters** **args** (*[str]*) – command line parameters as list of strings

**Returns** command line parameters namespace

**Return type** argparse.Namespace

**sl_window**()
Perform sliding window technique on the input the dataset.

**store**()
Save preprocessed data and the configuration of the preprocessor.

preprocessor.sliding_window.sliding_window.**run**(*args*)
Entry point for console_scripts

### Module contents

### preprocessor.standardizer package

### Submodules

### preprocessor.standardizer.standardizer module

This File contains the Standardizer class. To run this script uncomment or add the following lines in the [options.entry_points] section in setup.cfg:

**console_scripts =** standardizer = standardizer.__main__:main

Then run *python setup.py install* which will install the command *standardizer* inside your current environment.

**class** preprocessor.standardizer.standardizer.**Standardizer**(*conf*)
Bases: *preprocessor.preprocessor.Preprocessor*

The Standardizer preprocessor class

**core**()

**Core preprocessor task after starting the instance with the main method.** Decide from the arguments, what method to call.

Args: args (obj): command line parameters as objects

**load_from_config**()
Standardize the dataset from a config file.

**parse_args**(*args*)
Parse command line parameters

**Parameters** **args** (*[str]*) – command line parameters as list of strings

**Returns** command line parameters namespace

**Return type** argparse.Namespace

**standardize**()
Standardize the dataset.

**store**()
Save preprocessed data and the configuration of the preprocessor.

preprocessor.standardizer.standardizer.**run**(*args*)
Entry point for console_scripts

### Module contents

#### 1.4.1.2 Submodules

#### 1.4.1.3 preprocessor.conftest module

#### 1.4.1.4 preprocessor.preprocessor module

This File contains the Preprocessor class, it is the base class for DataTrimmer, FeatureSelector, Standardizer and SlidingWindow classes.

**class** preprocessor.preprocessor.**Preprocessor**(*conf*)
Bases: *preprocessor.preprocessor_base.PreprocessorBase*

Base class for DataTrimmer, FeatureSelector, Standardizer, SlidingWindow.

**assign_arguments**(*pargs*)

**core**()
Core preprocessor task after starting the instance with the main method. To be overriden by child classes depending on their preprocessor task.

**main**(*args*)

**Starts an instance. Main entry point allowing external calls.** Starts logging, parse command line arguments and start core.

Args: args ([str]): command line parameter list

**parse_args**(*args*)
Parse command line parameters, to be overriden by child classes depending on their command line parameters if they are console scripts.

Args: args ([str]): command line parameters as list of strings

Returns: argparse.Namespace: command line parameters namespace

**parse_cmd**(*parser*)

**store**()
Save preprocessed data and the configuration of the preprocessor.

#### 1.4.1.5 preprocessor.preprocessor_base module

This File contains the Preprocessor class, it is the base class for DataTrimmer, FeatureSelector, Standardizer, SlidingWindow.

**class** preprocessor.preprocessor_base.**PreprocessorBase**(*conf*)
Bases: object

Base class for Preprocessor.

**input_file = None**
Path of the input dataset

**load_ds**()
Save preprocessed data and the configuration of the preprocessor.

**output_file = None**
Path of the output dataset

**setup_logging**(*loglevel*)
Setup basic logging.

Args: loglevel (int): minimum loglevel for emitting messages

#### 1.4.1.6 Module contents

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## p

# Index

## R



## S



## T