pyoxynet

Release 0.0.8.1

Andrea Zignoli

Sep 24, 2023

CONTENTS

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

To use `pyoxynet`, first install it using pip:

```
(.venv) $ pip install pyoxynet
```

Note: Packages that require addition extra url cannot be installed via *setuptools*, which letely allows and suggests to use `pip` when possible. Therefore, as a work-around, TFLite is automatically installed with the following command the first time `pyoxynet` is imported.

The executed commend is:

```
(.venv) $ pip install --extra-index-url https://google-coral.github.io/py-repo/ tflite_
↪runtime
```

Currently, The TFLite installation process is completed with a line command inside Python, which i know is not the best solution.

1.1.2 Functions

To test the presence of the package you can use the `utilities.PrintHello()` function:

`utilities.`**`PrintHello`**`()`

This function prints to screen.

Args:
name (str): The name to use.

Returns:
none

1.1.3 Every-day functions

The optimal filter adopted for the project `utilities.optimal_filter(t, y, my_lambda)` function:

`utilities.`**`optimal_filter`**(*t*, *y*, *my_lambda*)

A bad ass optimisation filter

Parameters:

t
[array] Independent coord array

y
[array] Dependent coord array

my_lambda
[float] Smoothing factor

Returns:

x
[array] Filtered variable

`utilities.`**`normalize`**(*df*)

Pandas df normalisation

Parameters:
df (pd df) : input df

Returns:
result (pd df) : output df

`utilities.`**`random_walk`**(*length=1*, *scale_factor=1*, *variation=1*)

Random walk generator

Parameters:
length (int): Length of the output list scale_factor (float): Scale factor to be applied to the whole output variation (float): Local variation of the main signal with the random walk

Returns:
none

1.1.4 CPET data functions

`utilities.`**`load_csv_data`**(*csv_file='data_test.csv'*)

Loads data from csv file (returns test data if no arguments)

Parameters:
n_inputs (int) : Number of input variables. past_points (int) : Number of past inputs in the time-series.

Returns:
df (pandas df) : Model output example

1.1.5 TFLite model

To test if the TFLite model has been correctly initiated you can use `utilities.test_tfl_model(interpreter)` function:

`utilities.`**`test_tfl_model`**`()`

Test if the model is running correclty

Parameters:
interpreter (loaded tf.lite.Interpreter) : Loaded interpreter TFLite model

Returns:
x (array) : Model output example

`utilities.`**`load_tf_model`**`(`*n_inputs=7*, *past_points=40*`)`

This function loads the saved tflite models.

Args:
n_inputs (int): Number of input variables. past_points (int): Number of past inputs in the time-series.

Returns:
interpreter (tflite interpreter) : handle on the TFLite interpreter

`utilities.`**`load_tf_generator`**`()`

This function loads the saved tflite generator model.

Args:
None

Returns:
generator (tflite generator) : handle on the TFLite generator

`utilities.`**`pip_install_tflite`**`()`

Makes sure TFLite is installed by executing a pip install command from the command line (sub-optimal solution)

Parameters:
none

Returns:
none

1.1.6 Production functions

`utilities.`**`test_pyoxynet`**`(`*input_df=[]*, *n_inputs=7*, *past_points=40*`)`

Runs the pyoxynet inference

Parameters:
n_inputs (int) : Number of inputs (deafult to Oxynet configuration) past_points (int) : Number of past points in the time series (deafult to Oxynet configuration)

Returns:
x (array) : Model output example

`utilities.`**`create_probabilities`**`(`*duration=600*, *VT1=320*, *VT2=460*`)`

Creates the probabilities of being in different intensity domains

These probabilities are then sent to the CPET generator and they are used ot generate CPET vars that can replicate those probabilities

Parameters:

duration (int): Length of the test file VT1 (int): First ventilatory threshold, in time samples from the beginning of the test VT2 (int): Second ventilatory threshold, in time samples from the beginning of the test y_pm0 (double): This is used to set the first value of the probabilities. Indeed if you start from a higher VO2 you're not likely full in moderate domain (pm<1)

Returns:

p_mF (np array): Probability of being in the moderate intensity zone (-1:1) p_hF (np array): Probability of being in the heavy intensity zone (-1:1) p_sF (np array): Probability of being in the severe intensity zone (-1:1)

utilities.**generate_CPET**(*generator*, *plot=False*, *fitness_group=None*)

Actually generates the CPET file

Parameters:

length (int): Length of the output list fitness_group (int): Fitness level: low (1), medium (2), high (3). Default to random. noise_factor (float): Noise factor for white noise. Default to random.

Returns:

df (pd df): Pandas dataframe with CPET data included and ready to be processed by the model (if needed) data (dict): Data relative to the generated CPET

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX