Sklearn xarray Documentation

Release 0.0.0

Noah D Brenowitz

Nov 12, 2017

Contents

Contents:

CHAPTER 1

Installation

1.1 Stable release

To install Sklearn xarray, run this command in your terminal:

```
$ pip install sklearn_xarray
```

This is the preferred method to install Sklearn xarray, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for Sklearn xarray can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/nbren12/sklearn_xarray
```

Or download the tarball:

```
$ curl  -OL https://github.com/nbren12/sklearn_xarray/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

API Reference

2.1 Transformers

These classes implement the scikit-learn interface for transformations, and make working with xarray objects a breeze.

Select([key, sel])	
Stacker([feature_dims])	
XarrayUnion(transformer_list[, n_jobs, ...])	A version of feature union which keeps track of the index

2.1.1 sklearn_xarray.Select

class sklearn_xarray.**Select**(*key=None*, *sel=None*)

__init__(*key=None*, *sel=None*)

Methods

__init__([key, sel])	
fit(X[, y])	
fit_transform(X[, y])	Fit to data, then transform it.
get_params([deep])	Get parameters for this estimator.
inverse_transform(X)	
set_params(**params)	Set the parameters of this estimator.
transform(X)	

2.1.2 sklearn_xarray.Stacker

class sklearn_xarray.**Stacker**(*feature_dims=()*)

__init__(*feature_dims=()*)

Methods

__init__([feature_dims])	
fit(X[, y])	
fit_transform(X[, y])	Fit to data, then transform it.
get_params([deep])	Get parameters for this estimator.
inverse_transform(X)	Inverse transform
set_params(**params)	Set the parameters of this estimator.
transform(X: xarray.core.dataarray.DataArray)	

2.1.3 sklearn_xarray.XarrayUnion

class sklearn_xarray.**XarrayUnion**(*transformer_list*, *n_jobs=1*, *transformer_weights=None*)

A version of feature union which keeps track of the index

__init__(*transformer_list*, *n_jobs=1*, *transformer_weights=None*)

Methods

__init__(transformer_list[, n_jobs, ...])	
fit(X[, y])	Fit all transformers using X.
fit_transform(X[, y])	
get_feature_names()	Get feature names from all transformers.
get_params([deep])	Get parameters for this estimator.
set_params(**kwargs)	Set the parameters of this estimator.
transform(X)	Transform X separately by each transformer, concatenate results.

To see how to use these classes in conjuction with scikit-learn see *Linear Regression of multivariate data*.

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/nbren12/sklearn_xarray/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

3.1.4 Write Documentation

Sklearn xarray could always use more documentation, whether as part of the official Sklearn xarray docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/nbren12/sklearn_xarray/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.2 Get Started!

Ready to contribute? Here's how to set up *sklearn_xarray* for local development.

1. Fork the *sklearn_xarray* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/sklearn_xarray.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv sklearn_xarray
$ cd sklearn_xarray/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 sklearn_xarray tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/nbren12/sklearn_xarray/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ py.test tests.test_sklearn_xarray
```

CHAPTER 4

Credits

4.1 Development Lead

- Noah D Brenowitz <nbren12@gmail.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

5.1 0.0.0 (2017-11-08)

- First release on PyPI.

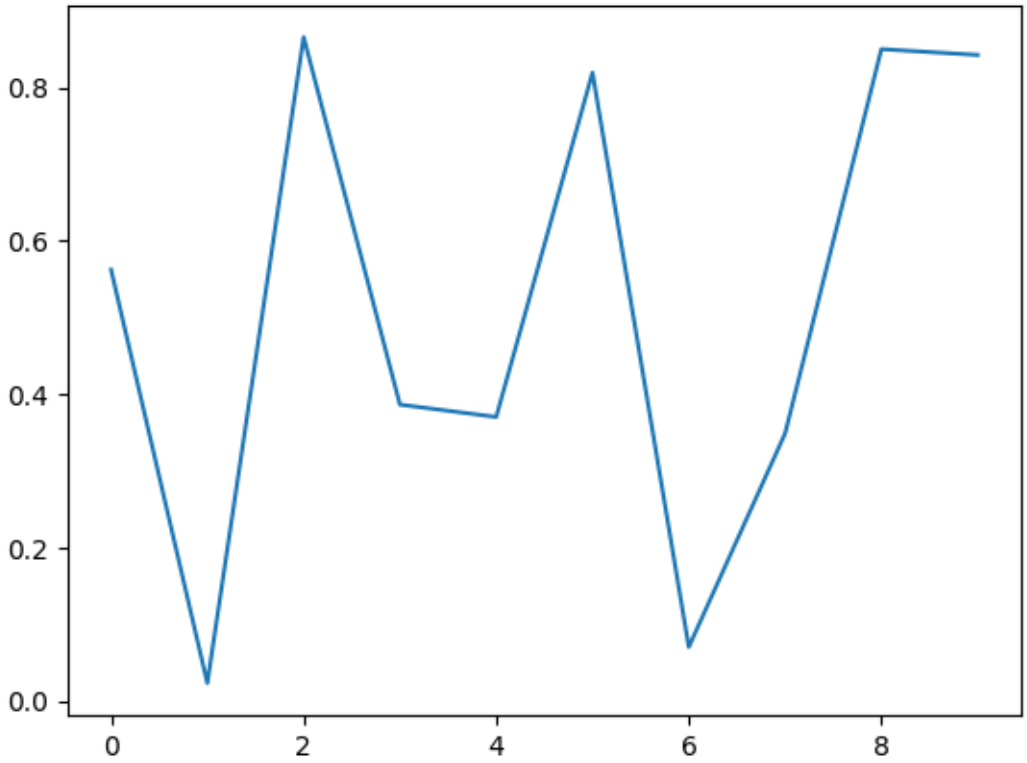
CHAPTER 6

Examples

These are examples of how to use this package.

6.1 Test of sphinx gallery

This is a test of sphinx gallery




```
import numpy as np
import matplotlib.pyplot as plt


plt.plot(np.random.rand(10))
```

Total running time of the script: (0 minutes 0.127 seconds)

6.2 Linear Regression of multivariate data

In this example, we demonstrate how to use sklearn_xarray classes to solve a simple linear regression problem on synthetic dataset.

This class demonstrates the use of *Stacker* and *Select*.

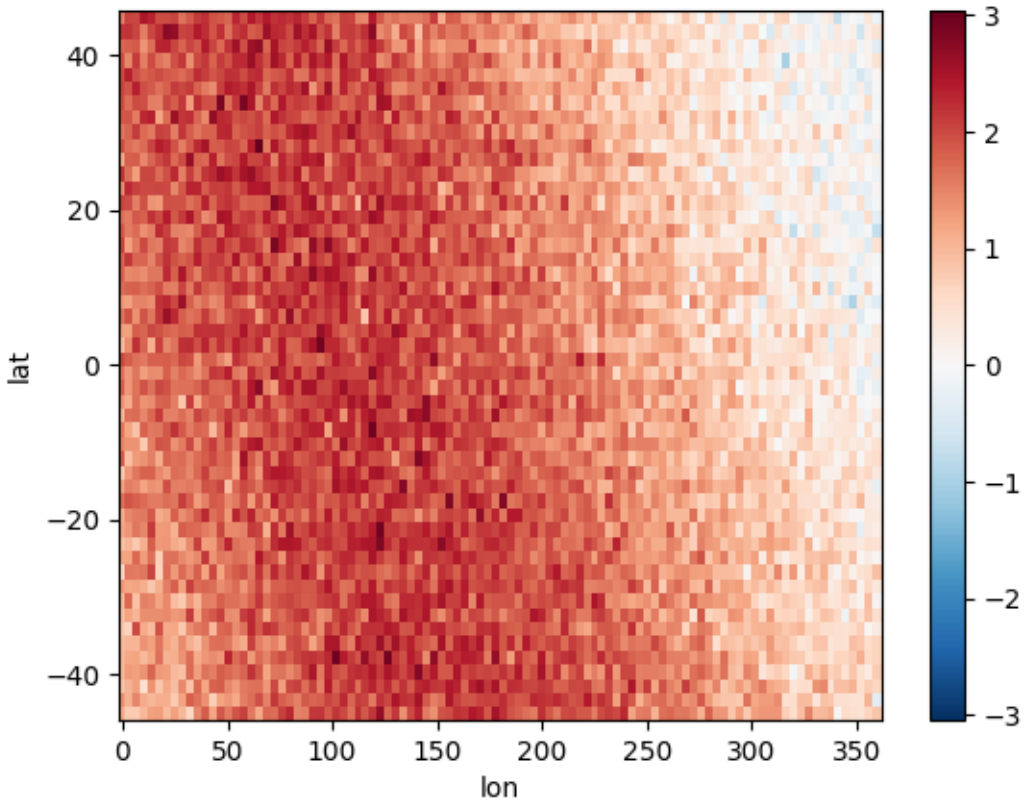
```
import numpy as np
import xarray as xr
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline, make_union

from sklearn_xarray import Stacker, Select

# Make synthetic data
lat, lon = np.ogrid[-45:45:50j, 0:360:100j]
```

```
noise = np.random.randn(lat.shape[0], lon.shape[1])

data_vars = {
    'a': (['lat', 'lon'], np.sin(lat/90 + lon/100)),
    'b': (['lat', 'lon'], np.cos(lat/90 + lon/100)),
    'noise': (['lat', 'lon'], noise)
}

coords = {'lat': lat.ravel(), 'lon': lon.ravel()}
dataset = xr.Dataset(data_vars, coords)
```

make a simple linear model for the output

$$y = a + .5b + 1$$

```
x = dataset[['a', 'b']]
y = dataset.a + dataset.b * .5 + .3 * dataset.noise  + 1

y.plot()
```



now we want to fit a linear regression model using these data

```
mod = make_pipeline(
    make_union(
        make_pipeline(Select('a'), Stacker()),
```

```
        make_pipeline(Select('b'), Stacker())),
    LinearRegression())
```

for now we have to use Stacker manually to transform the output data into a 2d array

```
y_np = Stacker().fit_transform(y)
print(y_np)
```

Out:

```
<xarray.DataArray (samples: 5000, features: 1)>
array([[ 1.138895],
       [ 0.799281],
       [ 0.790091],
       ...,
       [-0.134265],
       [ 0.388912],
       [-0.173836]])
Coordinates:
  * samples   (samples) MultiIndex
  - lat       (samples) float64 -45.0 -45.0 -45.0 -45.0 -45.0 -45.0 -45.0 ...
  - lon       (samples) float64 0.0 3.636 7.273 10.91 14.55 18.18 21.82 ...
  * features  (features) int64 1
```

fit the model

```
mod.fit(x, y_np)

# print the coefficients
lm = mod.named_steps['linearregression']
coefs = tuple(lm.coef_.flat)
print("The exact regression model is y = 1 + a + .5 b + noise")
print("The estimated coefficients are a: {}, b: {}".format(*coefs))
print("The estimated intercept is {}".format(lm.intercept_[0]))
```

Out:

```
The exact regression model is y = 1 + a + .5 b + noise
The estimated coefficients are a: 0.9826705586550489, b: 0.5070234156860342
The estimated intercept is 1.0154227436758414
```

Total running time of the script: (0 minutes 0.584 seconds)

CHAPTER 7

Usage

The best way to learn to use this package is to check out the *Examples*.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index

Symbols

S

X