nok / sklearn-porter

Transpile trained scikit-learn estimators to C, Java, JavaScript and others.

#scikit-learn #machine-learning #data-science #sklearn

532 commits | **3** branches | **11** releases | **4** contributors | MIT

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

nok Add SVC and NuSVC classifier — Latest commit `3ed7609` 8 hours ago

examples	Add new templates, examples and tests for the SVC and NuSVC classifiers	8 hours ago
scripts	Add start of local server	6 days ago
sklearn_porter	Add new templates, examples and tests for the SVC and NuSVC classifiers	8 hours ago
tests	Add new templates, examples and tests for the SVC and NuSVC classifiers	8 hours ago
.gitignore	Update and extend examples	10 days ago
.pylintrc	Improve code quality (Pylint)	7 months ago
.ruby-version	Upgrade Ruby to v2.3.4	5 months ago
.travis.yml	Fix blocking process	10 days ago
MANIFEST.in	Start changelog file and add it to the future releases	3 months ago
changelog.md	Add SVC and NuSVC classifier	8 hours ago
environment.yml	Remove package appnope	3 months ago
license.txt	Extend backwards compatibility by refactoring dependency handling, te…	10 months ago

readme.md	Add new templates, examples and tests for the SVC and NuSVC classifiers	8 hours ago
requirements.txt	Extend backwards compatibility by refactoring dependency handling, te…	10 months ago
setup.cfg	Add license file	10 months ago
setup.py	Refactor and fix lost requirements.txt file	3 months ago

readme.md

sklearn-porter

Transpile trained scikit-learn estimators to C, Java, JavaScript and others.
It's recommended for limited embedded systems and critical applications where performance matters most.

Machine learning algorithms

Algorithm	Programming language					
Classification	Java *	JS	C	Go	PHP	Ruby
svm.SVC	√, √ I	√	√		√	√
svm.NuSVC	√, √ I	√	√		√	√
svm.LinearSVC	√	√	√	√	√	√
tree.DecisionTreeClassifier	√, √ E, √ I	√, √ E	√, √ E	√, √ E	√, √ E	√, √ E
ensemble.RandomForestClassifier	√ E, √ I	√ E	√ E		√ E	√ E

ensemble.ExtraTreesClassifier	√ E, √ I	√ E	√ E		√ E	√ E
ensemble.AdaBoostClassifier	√ E, √ I	√ E, √ I	√ E			
neighbors.KNeighborsClassifier	√, √ I	√, √ I				
naive_bayes.GaussianNB	√	√				
naive_bayes.BernoulliNB	√	√				
neural_network.MLPClassifier	√, √ I	√				
Regression						
neural_network.MLPRegressor		√				

√ = is full-featured, E = with embedded model data, I = with imported model data, * = default language

Installation

```
$ pip install sklearn-porter
```

If you want the latest changes, you can install the module from the master branch:

```
$ pip uninstall -y sklearn-porter
$ pip install --no-cache-dir https://github.com/nok/sklearn-porter/zipball/master
```

Minimum requirements

```
- python>=2.7.3
- scikit-learn>=0.14.1
```

If you want to transpile a multilayer perceptron, you have to upgrade the scikit-learn package:

```
- python>=2.7.3
- scikit-learn>=0.18.0
```

Usage

Export

The following example shows how you can port a decision tree estimator to Java:

```
from sklearn.datasets import load_iris
from sklearn.tree import tree
from sklearn_porter import Porter

# Load data and train the classifier:
samples = load_iris()
X, y = samples.data, samples.target
clf = tree.DecisionTreeClassifier()
clf.fit(X, y)

# Export:
porter = Porter(clf, language='java')
output = porter.export(embed_data=True)
print(output)
```

The exported result matches the official human-readable version of the decision tree.

Prediction

Run the prediction(s) in the target programming language directly:

```
# ...
porter = Porter(clf, language='java')

# Prediction(s):
Y_java = porter.predict(X)
y_java = porter.predict(X[0])
y_java = porter.predict([1., 2., 3., 4.])
```

Integrity

Always compute and test the integrity between the original and the transpiled estimator:

```
# ...
porter = Porter(clf, language='java')

# Accuracy:
integrity = porter.integrity_score(X)
print(integrity)  # 1.0
```

Command-line interface

First of all have a quick view on the available arguments:

```
$ python -m sklearn_porter [-h] --input <PICKLE_FILE> [--output <DEST_DIR>] \
                           [--class_name <CLASS_NAME>] [--method_name <METHOD_NAME>] \
                           [--c] [--java] [--js] [--go] [--php] [--ruby] \
                           [--pipe]
```

The following example shows how you can save an trained estimator to the pickle format:

```
# ...

# Extract estimator:
joblib.dump(clf, 'estimator.pkl')
```

After that the estimator can be transpiled to JavaScript by using the following command:

```
$ python -m sklearn_porter -i estimator.pkl --js
```

The target programming language is changeable on the fly:

```
$ python -m sklearn_porter -i estimator.pkl --c
$ python -m sklearn_porter -i estimator.pkl --go
$ python -m sklearn_porter -i estimator.pkl --php
$ python -m sklearn_porter -i estimator.pkl --java
$ python -m sklearn_porter -i estimator.pkl --ruby
```

For further processing the argument `--pipe` can be used to pass the result:

```
$ python -m sklearn_porter -i estimator.pkl --js --pipe > estimator.js
```

For instance the result can be minified by using UglifyJS:

```
$ python -m sklearn_porter -i estimator.pkl --js --pipe | uglifyjs --compress -o estimator.min.js
```

Further information will be shown by using the `--help` argument:

```
$ python -m sklearn_porter --help
$ python -m sklearn_porter -h
```

Development

Environment

Install the required environment modules by executing the script environment.sh:

```
$ bash ./scripts/environment.sh
```

```
#!/usr/bin/env bash

conda env create -c conda-forge -n sklearn-porter python=2 -f environment.yml
source activate sklearn-porter
```

The following compilers or intepreters are required to cover all tests:

- GCC (`>=4.2`)
- Java (`>=1.6`)
- PHP (`>=7`)
- Ruby (`>=2.4.1`)
- Go (`>=1.7.4`)
- Node.js (`>=6`)

Testing

The tests cover module functions as well as matching predictions of transpiled estimators. Run all tests by executing the script test.sh:

```
$ bash ./scripts/test.sh
```

```
#!/usr/bin/env bash

python -m unittest discover -vp '*Test.py'
```

The test files have a specific pattern: `'[Algorithm][Language]Test.py'`:

```
$ python -m unittest discover -vp 'RandomForest*Test.py'
$ python -m unittest discover -vp '*JavaTest.py'
```

While you are developing new features or fixes, you can reduce the test duration by setting the number of tests:

```
$ N_RANDOM_FEATURE_SETS=15 N_EXISTING_FEATURE_SETS=30 python -m unittest discover -vp '*Test.py'
```

Quality

It's highly recommended to ensure the code quality. For that I use Pylint, which you can run by executing the script lint.sh:

```
$ bash ./scripts/lint.sh
```

```
#!/usr/bin/env bash

find ./sklearn_porter -name '*.py' -exec pylint {} \;
```

Citation

If you use this implementation in you work, please add a reference/citation to the paper. You can use the following BibTeX entry:

```
@misc{SkPoDaMo,
  author = {Darius Morawiec},
  title = {sklearn-porter: Transpile trained scikit-learn estimators to C, Java, JavaScript and others},
  url = {https://github.com/nok/sklearn-porter},
  year = {2016--2017}
}
```

License

The module is Open Source Software released under the MIT license.

Questions?

Don't be shy and feel free to contact me on Twitter or Gitter.