Feature Selection GA

Kaushal Shetty

Sep 29, 2020

INSTALLATION USAGE

CHAPTER

ONE

INSTALLATION

The package is published on PyPI and can be installed with `pip` (or any equivalent):

```
pip install feature-selection-ga
```

CHAPTER

TWO

USAGE

To use this package, import it:

```
from sklearn.datasets import make_classification
from sklearn import linear_model
from feature_selection_ga import FeatureSelectionGA, FitnessFunction
X, y = make_classification(n_samples=100, n_features=15, n_classes=3,
                           n_informative=4, n_redundant=1, n_repeated=2,
                           random_state=1)

model = linear_model.LogisticRegression(solver='lbfgs', multi_class='auto')
fsga = FeatureSelectionGA(model,X,y, ff_obj = FitnessFunction())
pop = fsga.generate(100)

#print(pop)
```

TODO: Document usage

CHAPTER

THREE

CHANGELOG

3.1 v0.1.2 (2020-09-29)

Full Changelog

3.2 v0.1.1 (2020-09-29)

Full Changelog

** This Changelog was automatically generated by github_changelog_generator*

CHAPTER

FOUR

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/kaushalshetty/FeatureSelectionGA/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/kaushalshetty/FeatureSelectionGA/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/FeatureSelectionGA.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 FeatureSelectionGA
$ cd FeatureSelectionGA/
$ 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 python_package tests -->
$ pytest
<!-- $ tox -->
```

   To get flake8 and tox, 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.

4.3 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 3.6, 3.7 and 3.8. Check the build and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ pytest tests
```

4.5 Publish to PyPI

A reminder for the maintainers on how to publish to PyPI. Make sure all your changes are committed, then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

This will create a tag, update changelog and draft Github release for you. Once all the tests have passed, publish the release on Github by pointing at the tag you just pushed. This will trigger the upload of the release to PyPI.

CHAPTER

FIVE

FEATURESELECTIONGA

5.1 Feature Selection using Genetic Algorithm (DEAP Framework)

Data scientists find it really difficult to choose the right features to get maximum accuracy especially if you are dealing with a lot of features. There are currenlty lots of ways to select the right features. But we will have to struggle if the feature space is really big. Genetic algorithm is one solution which searches for one of the best feature set from other features in order to attain a high accuracy.

5.1.1 Installation:

```
$ pip install feature-selection-ga
```

5.1.2 Usage:

```
from sklearn.datasets import make_classification
from sklearn import linear_model
from feature_selection_ga import FeatureSelectionGA, FitnessFunction

X, y = make_classification(n_samples=100, n_features=15, n_classes=3,
                           n_informative=4, n_redundant=1, n_repeated=2,
                           random_state=1)

model = linear_model.LogisticRegression(solver='lbfgs', multi_class='auto')
fsga = FeatureSelectionGA(model,X,y, ff_obj = FitnessFunction())
pop = fsga.generate(100)

#print(pop)
```

5.1.3 Usage (Advanced):

By default, the FeatureSelectionGA has its own fitness function class. We can also define our own FitnessFunction class.

```
class FitnessFunction:
    def __init__(self,n_splits = 5,*args,**kwargs):
        """
            Parameters
            -----------
            n_splits :int,
                Number of splits for cv

            verbose: 0 or 1
        """
        self.n_splits = n_splits

    def calculate_fitness(self,model,x,y):
        pass
```

With this, we can design our own fitness function by defining our calculate fitness! Consider the following example from Vieira, Mendoca, Sousa, et al. (2013) `$f(X) = \alpha(1-P) + (1-\alpha) \left(1 - \dfrac{N_f}{N_t}\right)$`

Define the constructor **init** with needed parameters: alpha and N_t.

```
class FitnessFunction:
    def __init__(self,n_total_features,n_splits = 5, alpha=0.01, *args,**kwargs):
        """
            Parameters
            -----------
            n_total_features :int
                    Total number of features N_t.
            n_splits :int, default = 5
                Number of splits for cv
            alpha :float, default = 0.01
                Tradeoff between the classifier performance P and size of
                feature subset N_f with respect to the total number of features
                N_t.

            verbose: 0 or 1
        """
        self.n_splits = n_splits
        self.alpha = alpha
        self.n_total_features = n_total_features
```

Next, we define the fitness function, the name has to be calculate_fitness:

```
    def calculate_fitness(self,model,x,y):
        alpha = self.alpha
        total_features = self.n_total_features

        cv_set = np.repeat(-1.,x.shape[0])
        skf = StratifiedKFold(n_splits = self.n_splits)
        for train_index,test_index in skf.split(x,y):
            x_train,x_test = x[train_index],x[test_index]
            y_train,y_test = y[train_index],y[test_index]
```

(continues on next page)

(continued from previous page)

```
            if x_train.shape[0] != y_train.shape[0]:
                raise Exception()
            model.fit(x_train,y_train)
            predicted_y = model.predict(x_test)
            cv_set[test_index] = predicted_y

        P = accuracy_score(y, cv_set)
        fitness = (alpha*(1.0 - P) + (1.0 - alpha)*(1.0 - (x.shape[1])/total_
↪features))
        return fitness
```

Example: You may also see `example2.py`

```
X, y = make_classification(n_samples=100, n_features=15, n_classes=3,
n_informative=4, n_redundant=1, n_repeated=2,
random_state=1)

# Define the model

model = linear_model.LogisticRegression(solver='lbfgs', multi_class='auto')

# Define the fitness function object

ff = FitnessFunction(n_total_features= X.shape[1], n_splits=3, alpha=0.05)
fsga = FeatureSelectionGA(model,X,y, ff_obj = ff)
pop = fsga.generate(100)
```

Example adopted from pyswarms