期末報告--
[Lasso Regression]

學號：R06525065
姓名：王志明

演算法文件(http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html)

Outline

Algorithm Introduction

Code Review

Model Preview

Live Demo(Using InAnalysis)

Conclusion

Reference

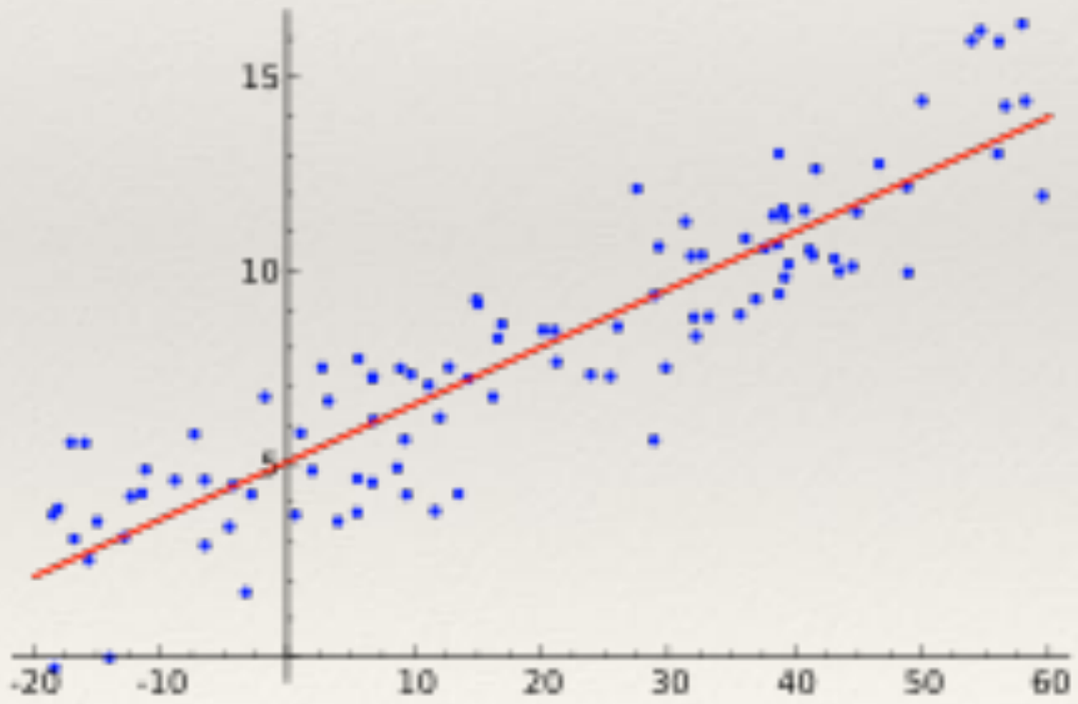
Algorithm Introduction

Linear regression:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

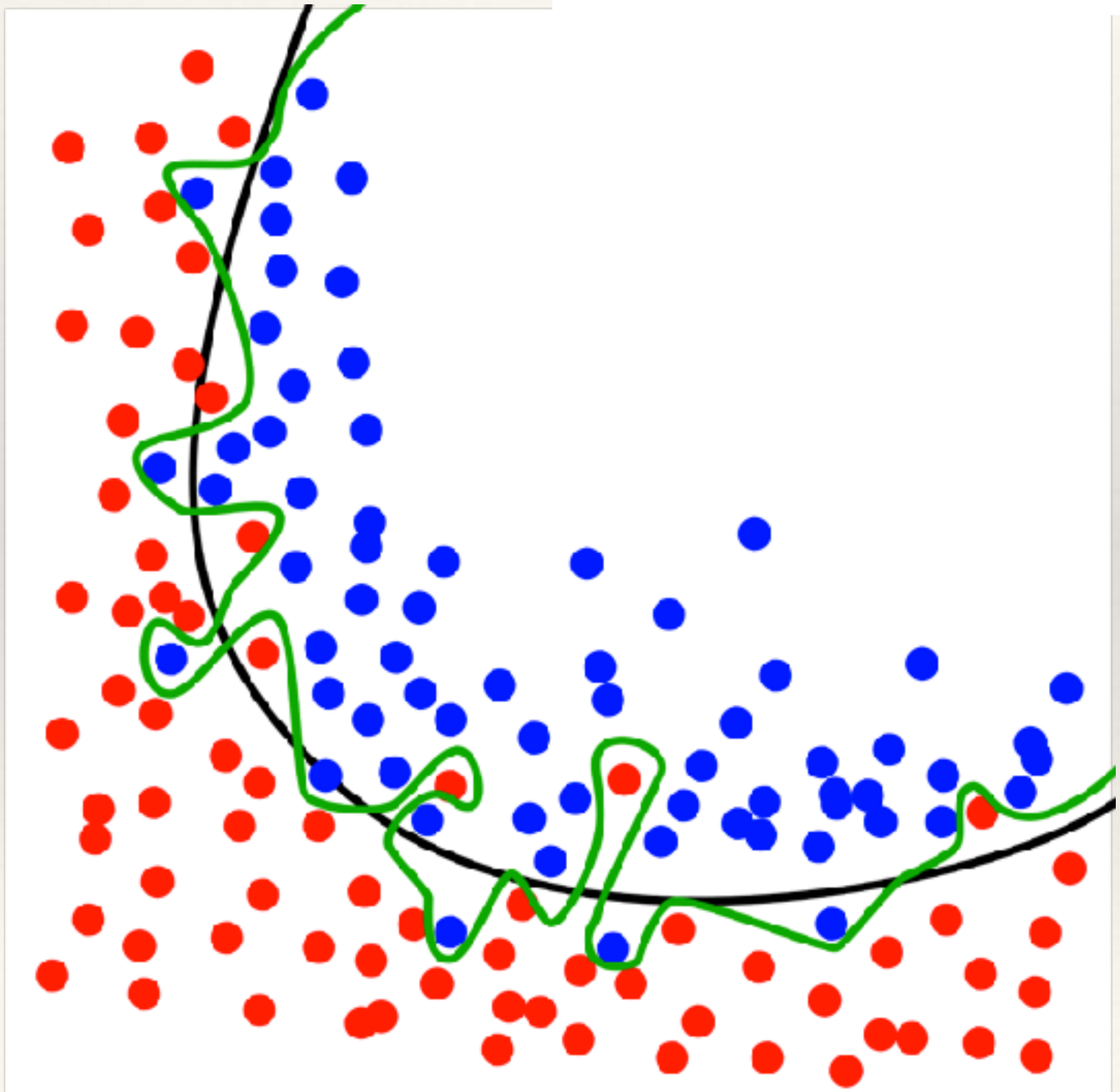
Cost :

$$RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2$$



Overfitting

$$min_{\beta_0,\cdots,\beta_k} \sum_{i=1}^{n} [Y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{ik})]^2 + \gamma \sum_{j=0}^{k} |\beta_j|$$



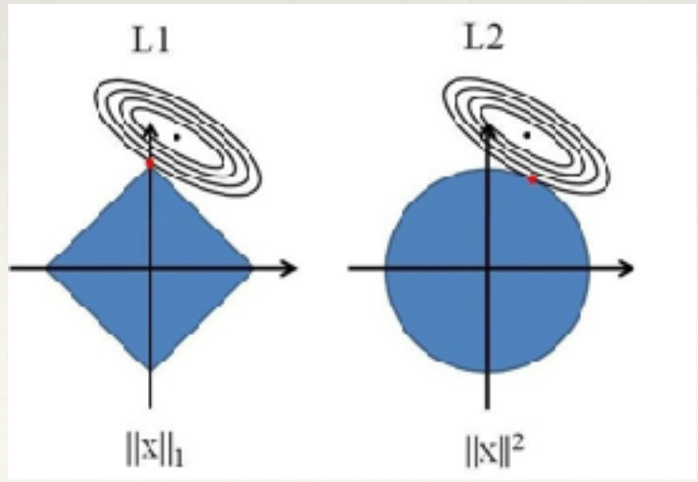
Algorithm Introduction

Lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.

$$(\hat{\alpha}, \hat{\beta}) = \arg\min\left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 \right\} \qquad \text{subject to } \sum_j |\beta_j| \leqslant t.$$

$$min_{\beta_0,\cdots,\beta_k} \sum_{i=1}^{n} \left[Y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}) \right]^2 + \gamma \sum_{j=0}^{k} |\beta_j|$$

p=2



Code Review

algo_component.py

algo_component.py

```python
from abc import abstractmethod
import logging

logging.basicConfig(level=logging.DEBUG)
log = logging.getLogger(__name__)


class ParamsDefinition:
    def __init__(self, name, type, range, default_value, description):
        self.name = name
        self.type = type
        self.range = range
        self.default_value = default_value
        self.description = description

    def get_params_definition(self):
        return self.__dict__


class ParamsDefinitionSet:
    def __init__(self):
        self.params_definition_set = {}
        raise NotImplementedError

    def get_params_definition_json_list(self):
        definition_set_json_list = []
        for params_object in self.params_definition_set:
            definition_set_json_list.append(params_object.get_params_definition())
        return definition_set_json_list


# Inanalysis Algorithm interface
class InanalysisAlgo(ABC):
    @abstractmethod
    def do_algo(self, input_params):
        raise NotImplementedError

    def check_input_params(self, input_params_definition, user_input_params):
```

Code Review

Lasso_r06525065.py — UNREGISTERED

Lasso_r06525065.py ×

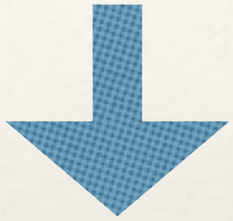
```python
class ParamsDefinitionSet(alc.ParamsDefinitionSet):
    def __init__(self):
        self.params_definition_set =\
            {
                alc.ParamsDefinition(name='alpha', type='float', range='0.0, 1.0', default_value='1.0', description=''),
                alc.ParamsDefinition(name='fit_intercept', type='boolean', range='True,False', default_value='False', description=''),
                alc.ParamsDefinition(name='normalize', type='boolean', range='True,False', default_value='False', description=''),
                alc.ParamsDefinition(name='copy_X', type='boolean', range='True,False', default_value='True', description=''),
                alc.ParamsDefinition(name='max_iter', type='int', range='', default_value='-1', description=''),

            }


class Lasso_r06525065(alc.InanalysisAlgo):
    def __init__(self):
        self.input_params_definition = ParamsDefinitionSet()

    def get_input_params_definition(self):
        return self.input_params_definition.get_params_definition_json_list()

    def do_algo(self, input):
        control_params = input.algo_control.control_params
        if not self.check_input_params(self.get_input_params_definition(), control_params):
            log.error("Check input params type error.")
            return None
        mode = input.algo_control.mode
        data = input.algo_data.data
        label = input.algo_data.label
        if mode == 'training':
            try:
                model = linear_model.Lasso(
                    alpha=control_params["alpha"],
                    normalize=control_params["normalize"],
                    copy_X=control_params["copy_X"],
                    max_iter=control_params["max_iter"],
                    fit_intercept=control_params['fit_intercept'],
                )
                model.fit(X=data, y=label)
                algo_output = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': ''},
                                            algo_data={'data': data, 'label': label},
                                            algo_model={'model_params': model.get_params(), 'model_instance': model})
            except Exception as e:
```

Line 1, Column 1 — Spaces: 4 — Python

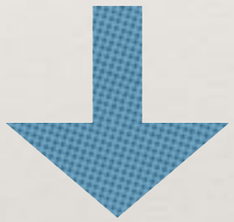
Model Preview

```
arg_dict = {
    "fit_intercept": True,
    "normalize": False,
    "copy_X": True,
    "alpha": 1.0,
    "max_iter": 100,
}
```

There is 13 inputs if we use Boston data.



It hard to use 3D diagram to perform the prediction.



Difference between prediction and real Boston_data's target.

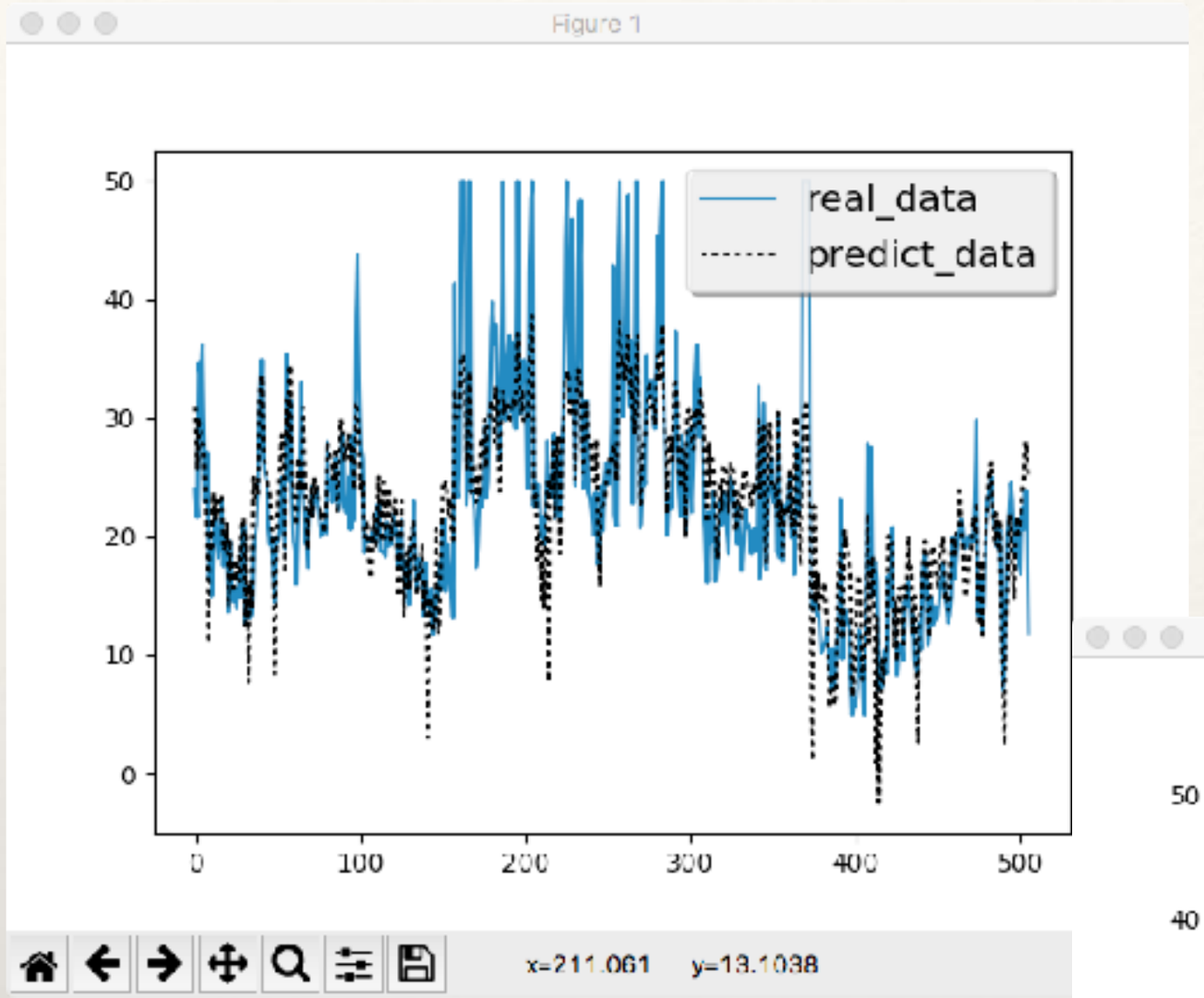


Figure 1
real_data
predict_data
50
40
30
20
10
0
0
100
200
300
400
500
x=211.061 y=13.1038

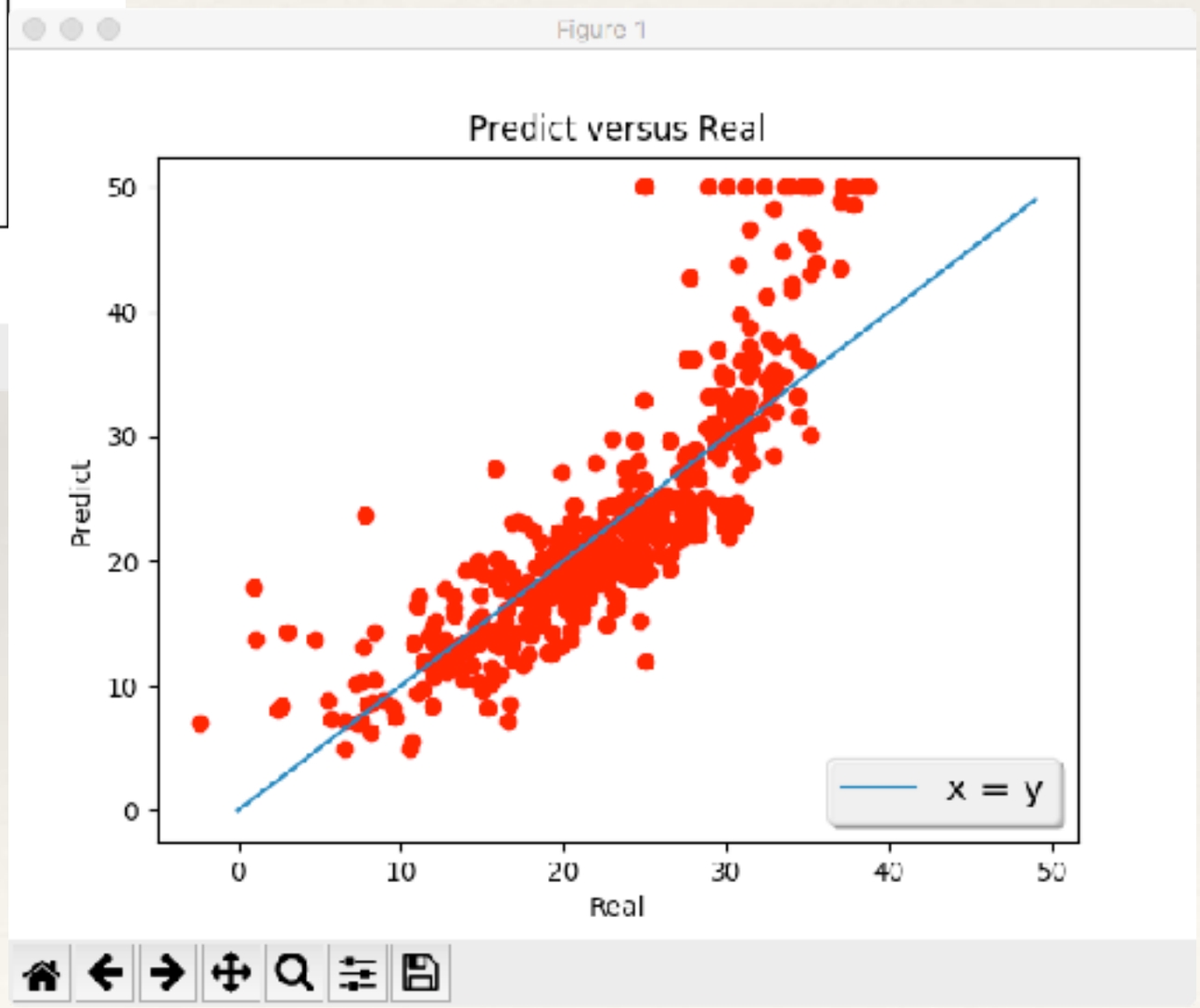
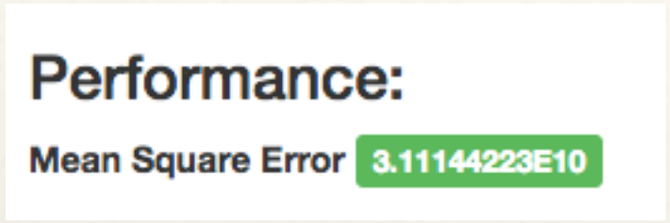


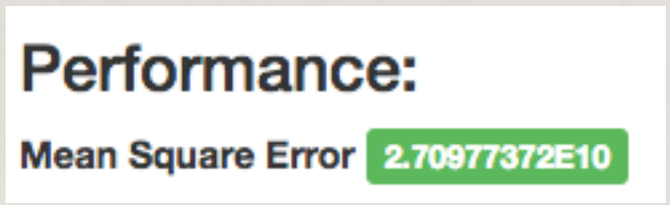
Figure 1
Predict versus Real
Predict
Real
x = y
0
10
20
30
40
50

Midterm:(linear regression)



linear-regression , mean square error: 20.7471433603

Final:(Lasso regression)



Lasso_r06525065 , mean square error: 25.399124341

Conclusion

- 實作machine learning的程式
- 參與團體合作
- 要試多種方法

Reference

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html