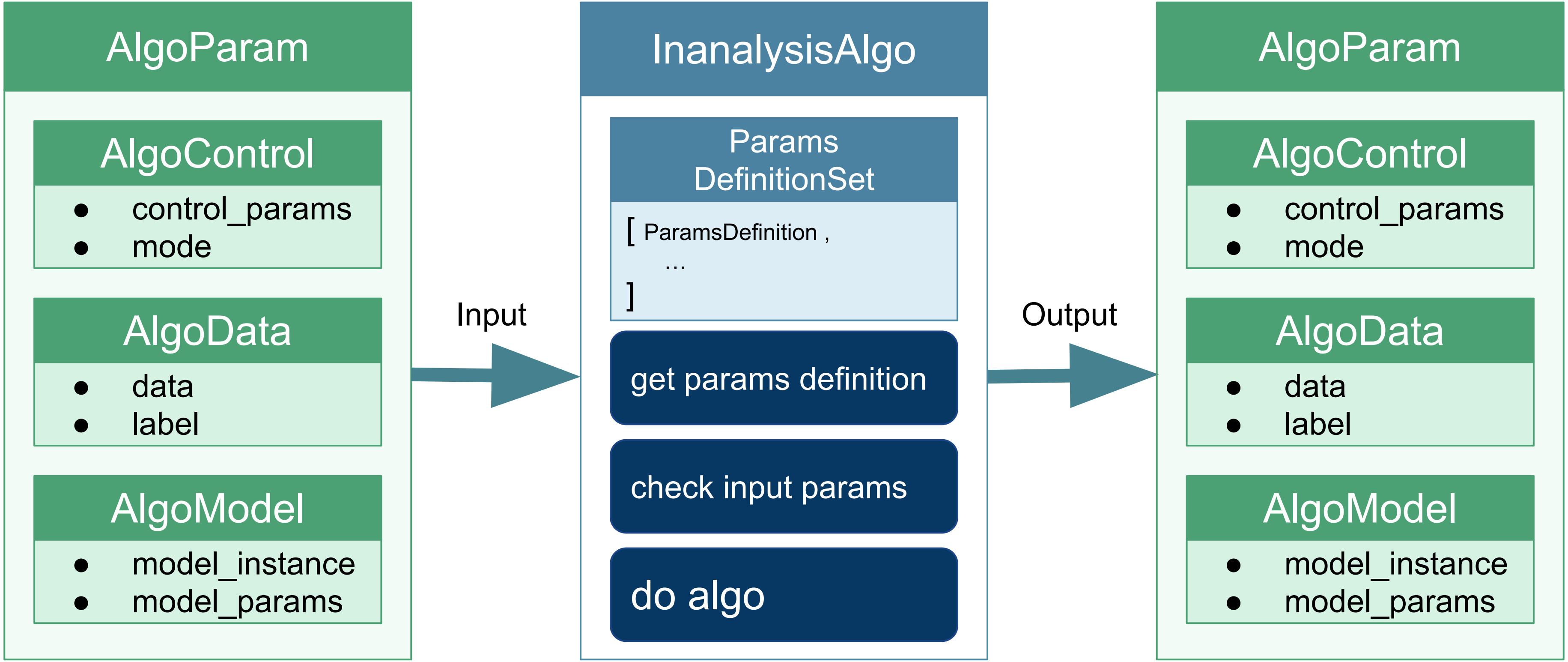
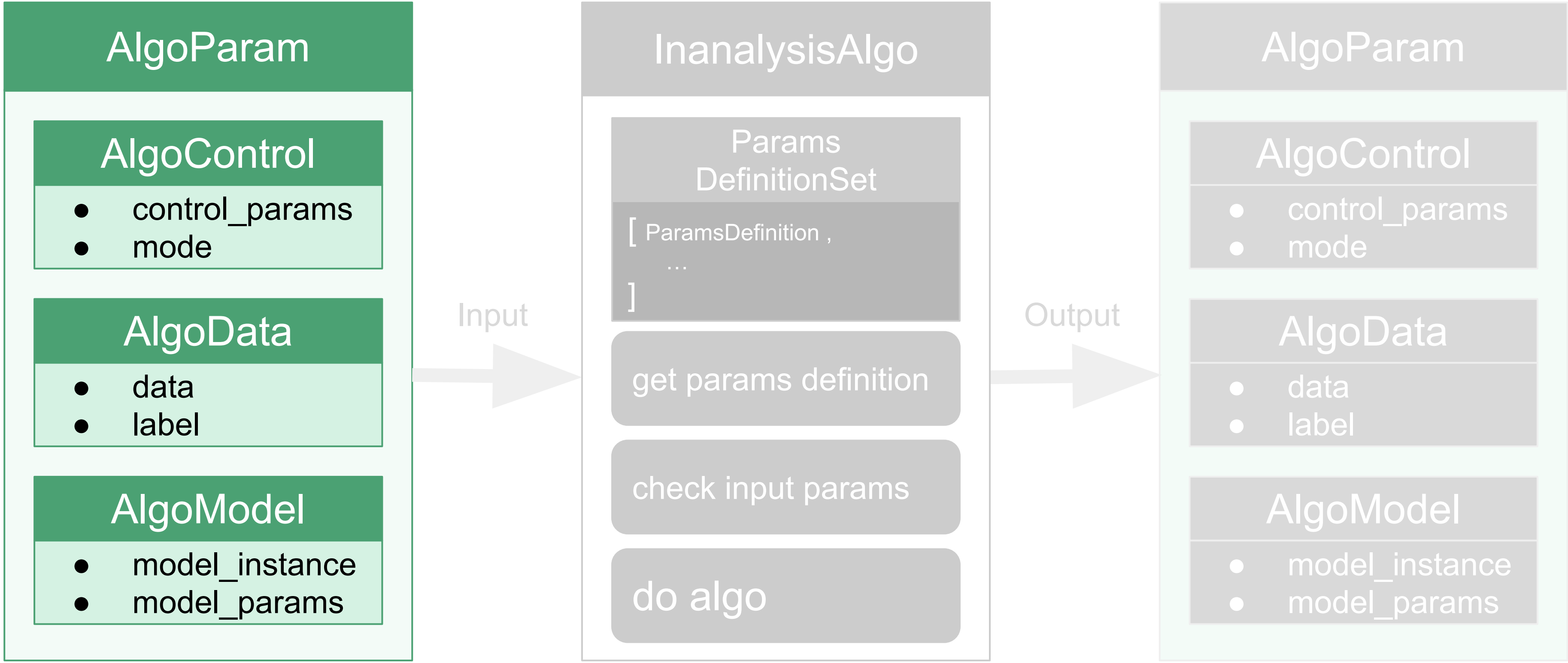
InAnalysis演算法模組介紹

InAnalysis演算法模組 架構



InAnalysis演算法模組 流程：建立輸入物件



AlgoParam

AlgoControl

- control_params
- mode

AlgoData

- data
- label

AlgoModel

- model_instance
- model_params

Input

InanalysisAlgo

Params DefinitionSet

[ParamsDefinition ,
…
]

get params definition

check input params

do algo

Output

AlgoParam

AlgoControl

- control_params
- mode

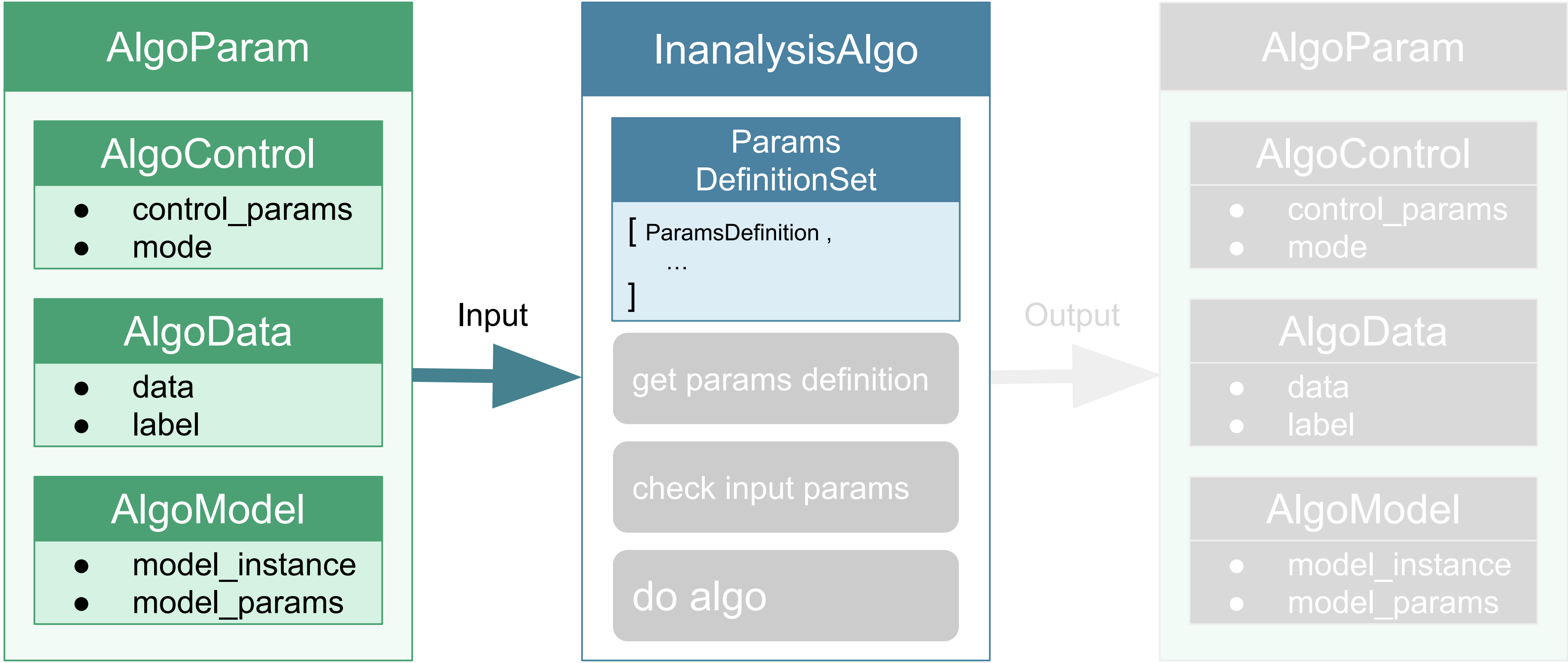
AlgoData

- data
- label

AlgoModel

- model_instance
- model_params

InAnalysis演算法模組 流程:建立演算法物件



AlgoParam

AlgoControl
- control_params
- mode

AlgoData
- data
- label

AlgoModel
- model_instance
- model_params

Input

InanalysisAlgo

Params DefinitionSet

[ParamsDefinition ,
…
]

get params definition

check input params

do algo

Output

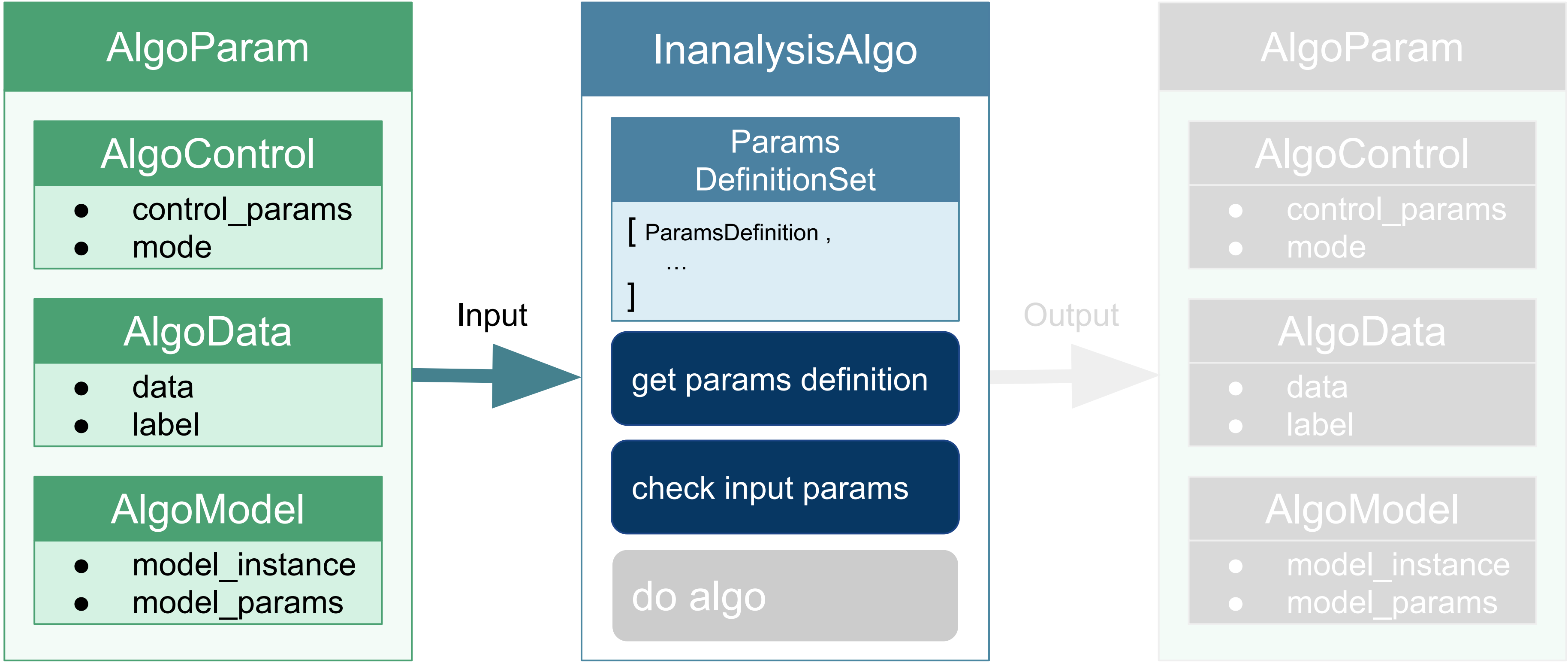
AlgoParam

AlgoControl
- control_params
- mode

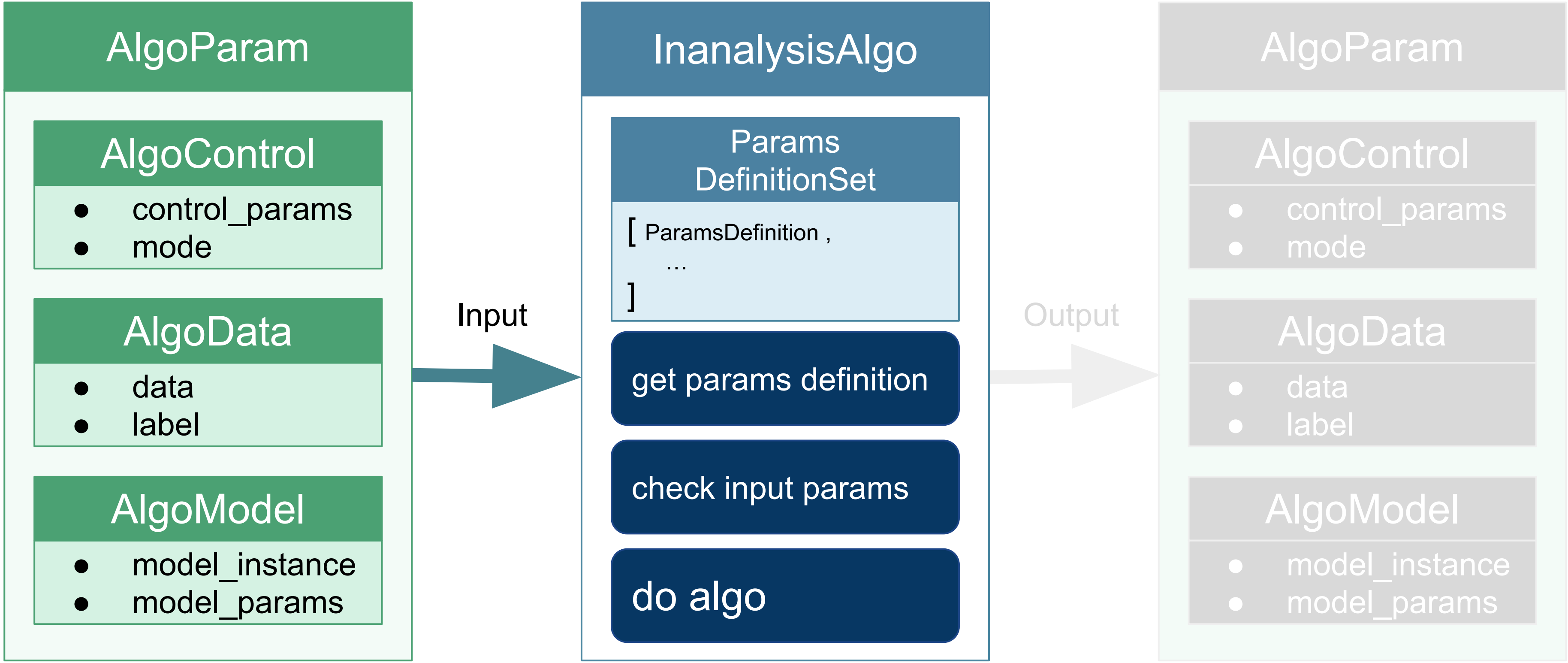
AlgoData
- data
- label

AlgoModel
- model_instance
- model_params

InAnalysis演算法模組 流程：檢查輸入參數



InAnalysis演算法模組 流程：執行演算法



AlgoParam

AlgoControl
- control_params
- mode

AlgoData
- data
- label

AlgoModel
- model_instance
- model_params

Input

InanalysisAlgo

Params DefinitionSet

[ParamsDefinition ,
…
]

get params definition

check input params

do algo

Output

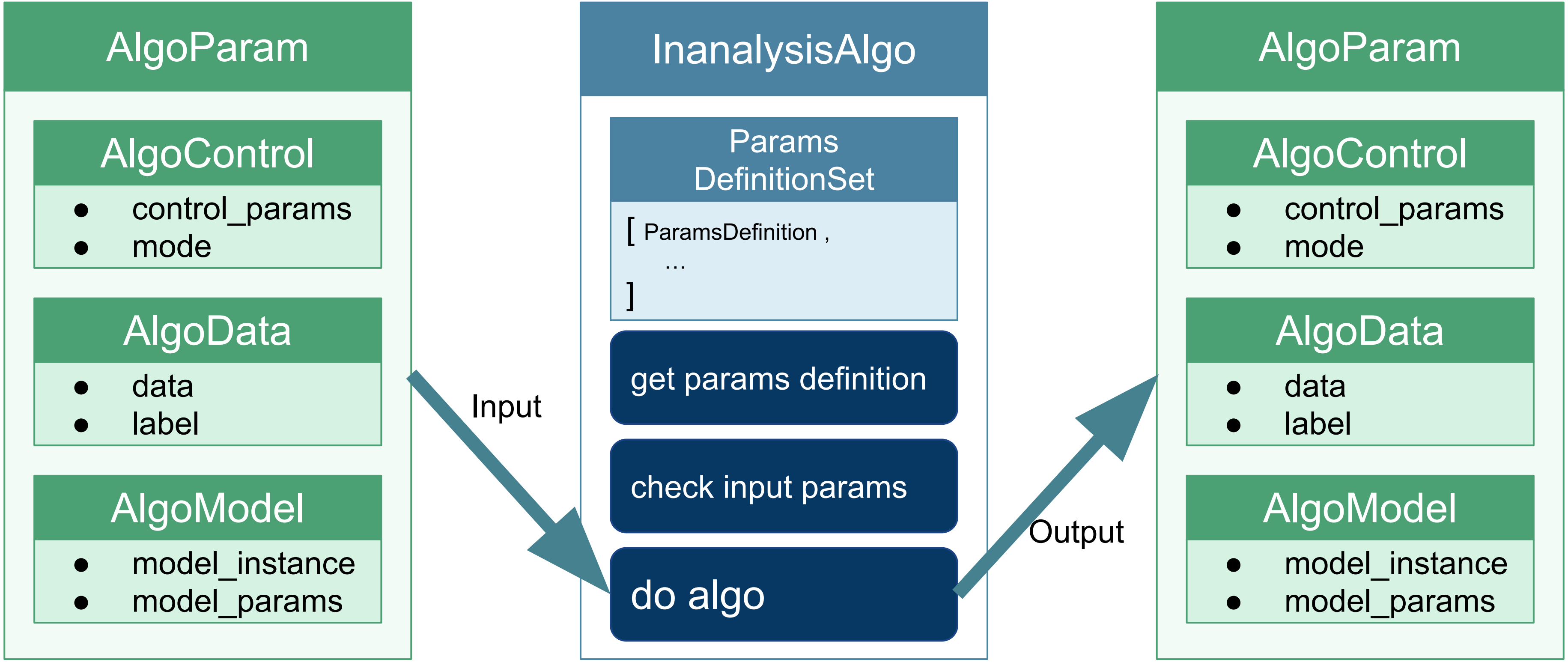
AlgoParam

AlgoControl
- control_params
- mode

AlgoData
- data
- label

AlgoModel
- model_instance
- model_params

InAnalysis演算法模組 流程：得到輸出物件



AlgoParam

AlgoControl

- control_params
- mode

AlgoData

- data
- label

AlgoModel

- model_instance
- model_params

Input

InanalysisAlgo

Params DefinitionSet

[ParamsDefinition ,
 …
]

get params definition

check input params

do algo

Output

AlgoParam

AlgoControl

- control_params
- mode

AlgoData

- data
- label

AlgoModel

- model_instance
- model_params

概覽

Project 2 需要實作 2大部份

1.演算法模組實作 (10 points)

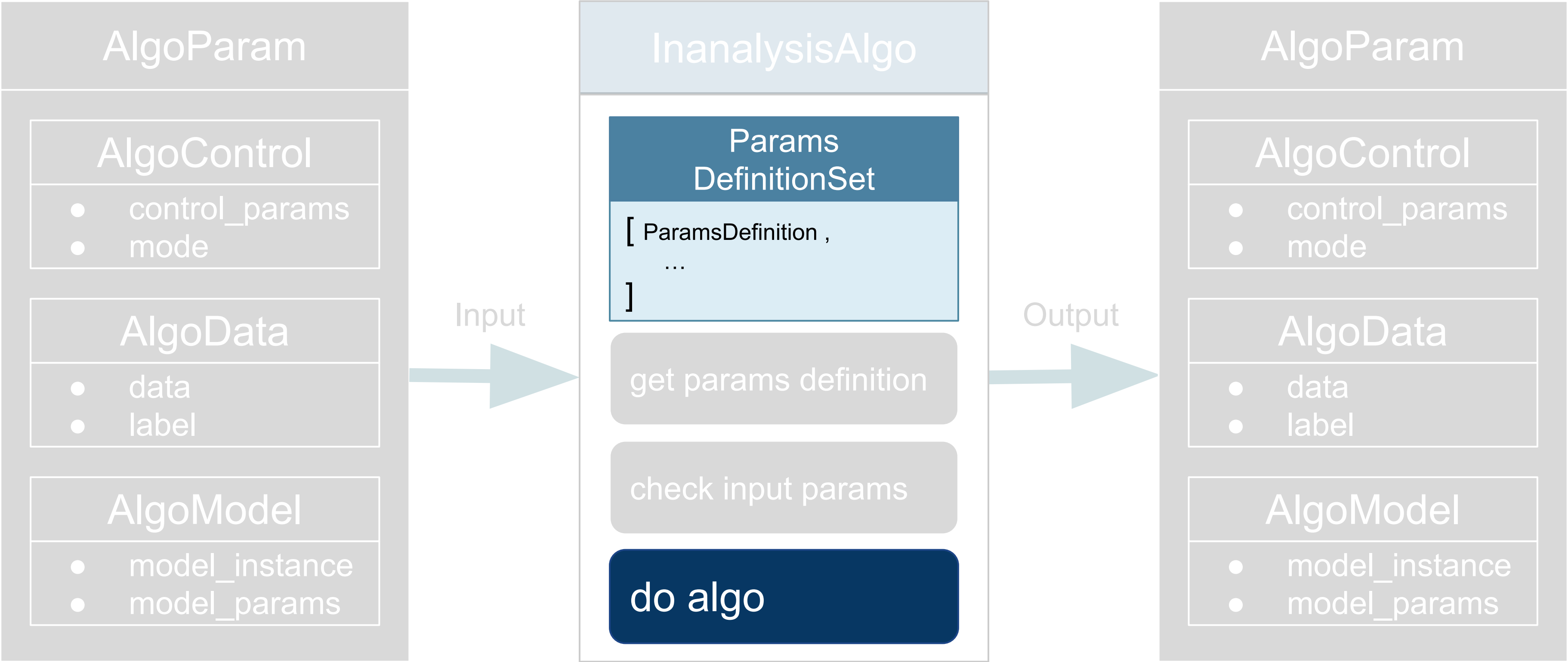
- 專案類別_資料夾
    - **演算法名稱_學號.py**
        - 演算法參數定義
          : ParamsDefinitionSet
        - 演算法函式 : do_algo
- utils.py
    - 演算法工具

2.演算法測試實作 (15 points)

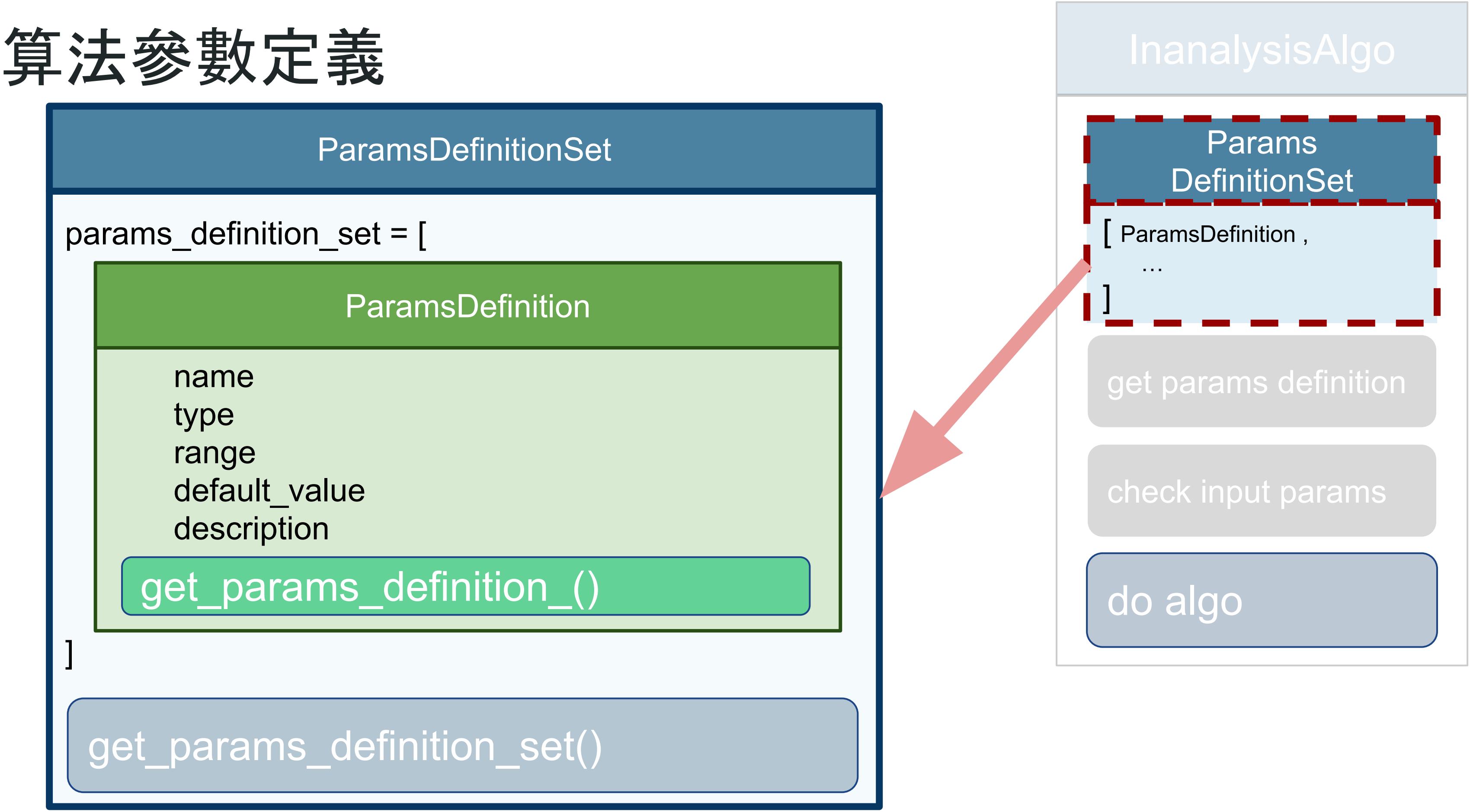
- tests_資料夾
    - **test_演算法名稱_學號.py**

1. 演算法模組實作

InAnalysis演算法模組 實作說明



演算法參數定義



ParamsDefinitionSet

params_definition_set = [

ParamsDefinition

name
type
range
default_value
description

get_params_definition_()

]

get_params_definition_set()

InanalysisAlgo

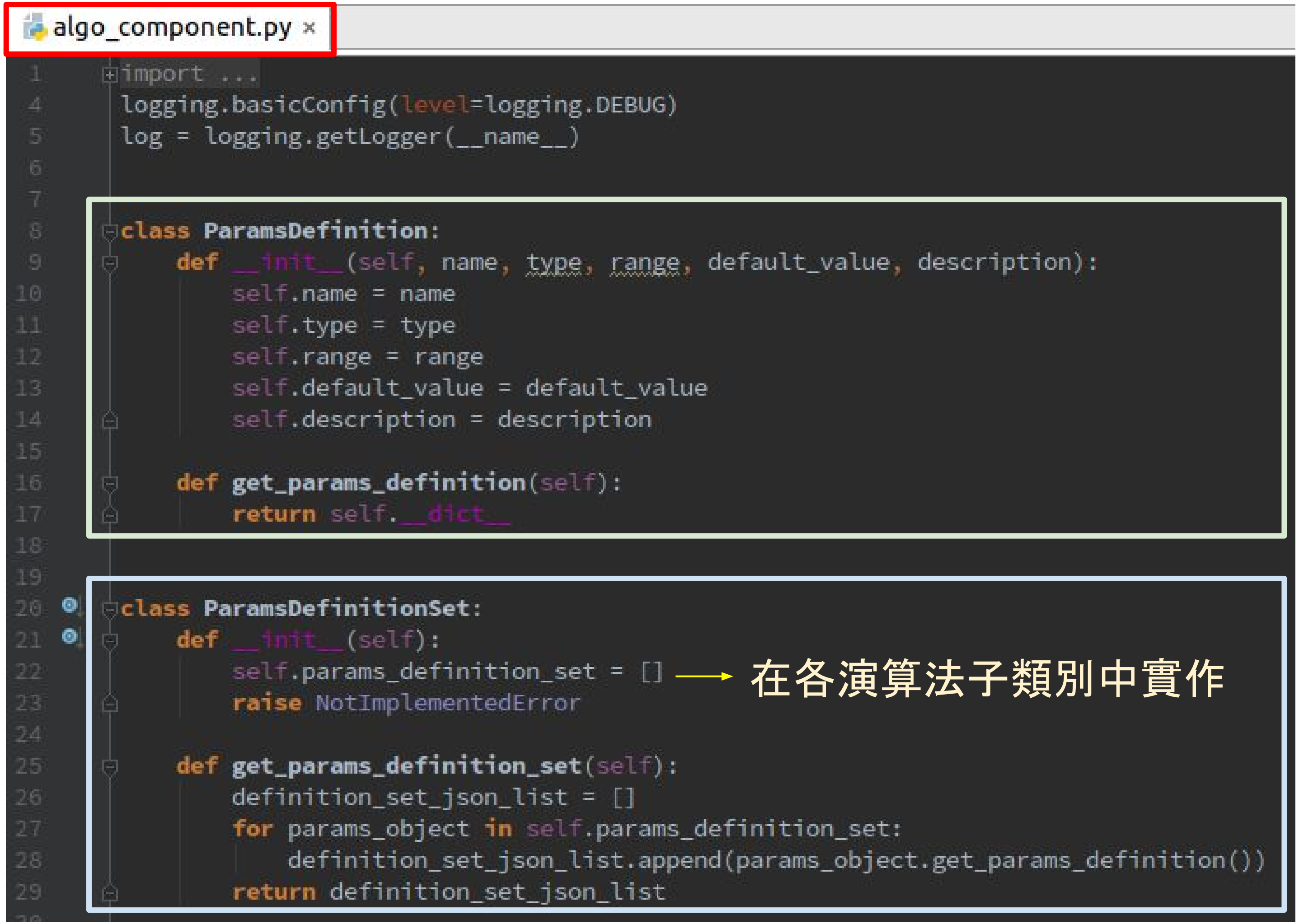
Params DefinitionSet

[ParamsDefinition ,
…
]

get params definition

check input params

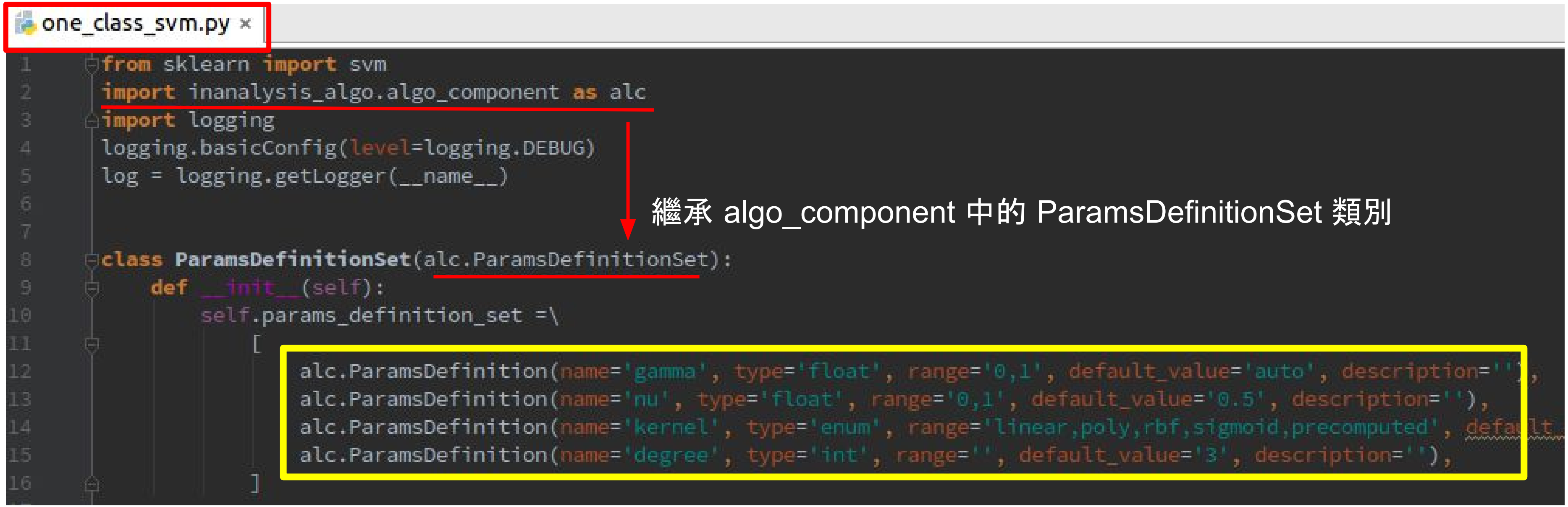
do algo



algo_component.py

```python
import ...
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_set(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
```

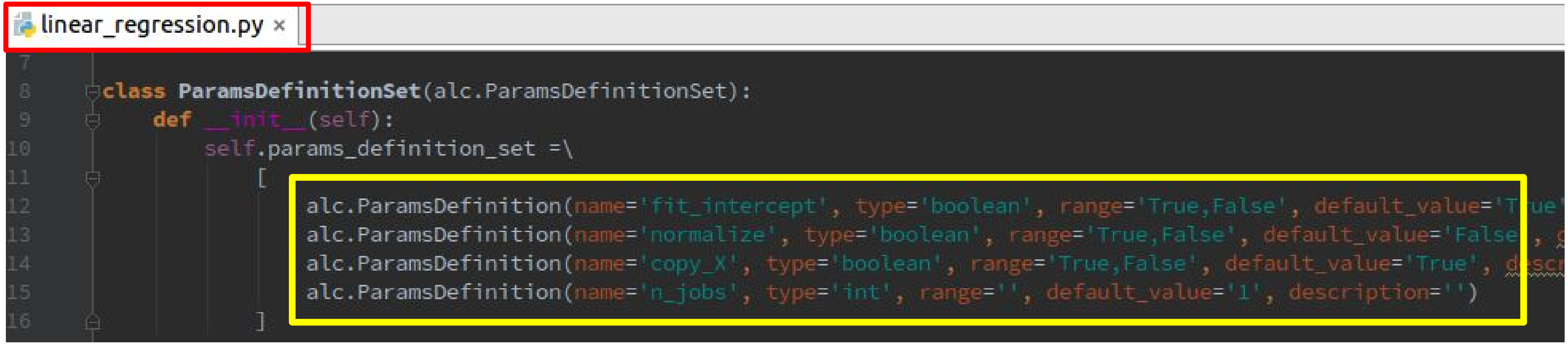
在各演算法子類別中實作



one_class_svm.py

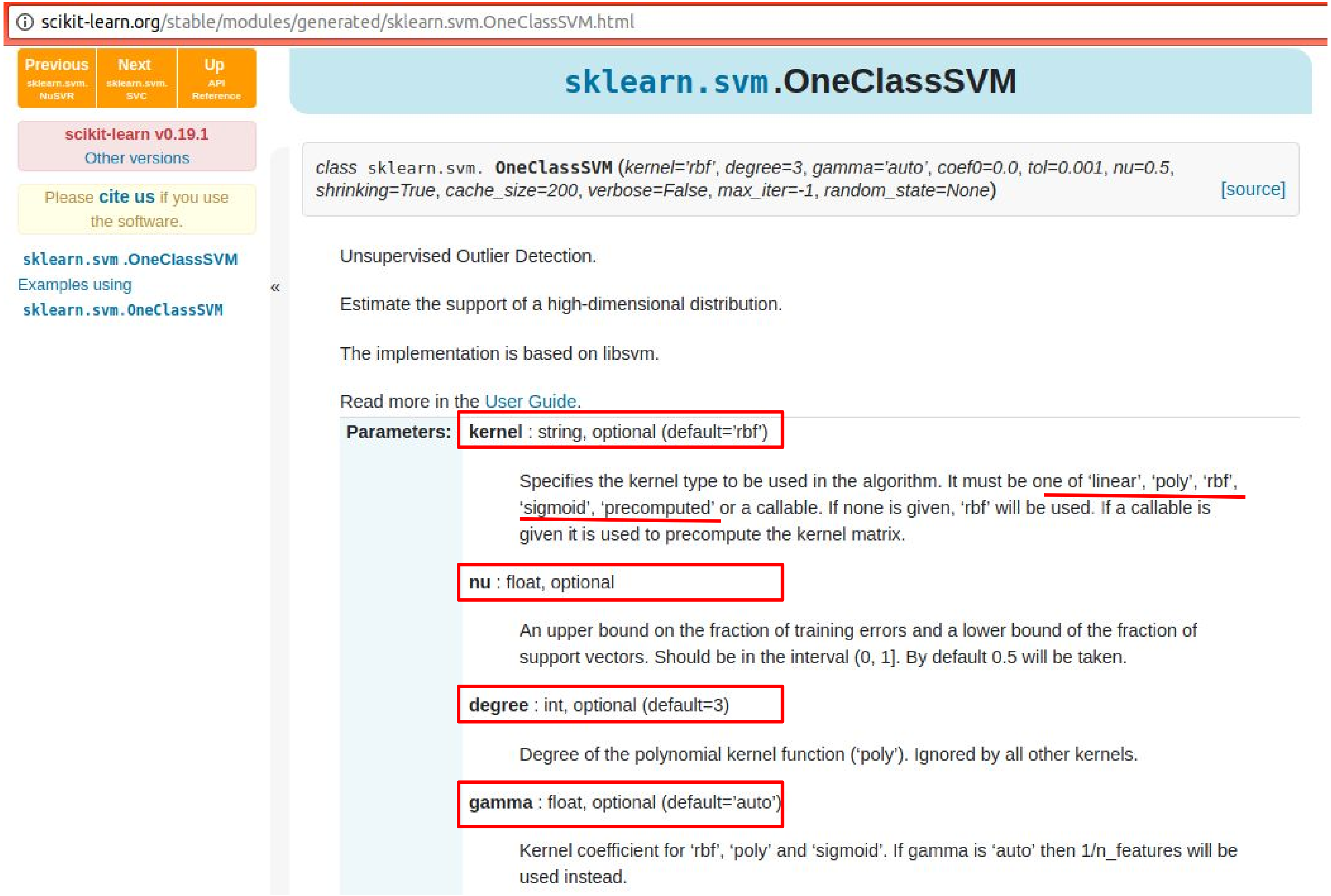
```python
from sklearn import svm
import inanalysis_algo.algo_component as alc
import logging
logging.basicConfig(level=logging.DEBUG)
log = logging.getLogger(__name__)


class ParamsDefinitionSet(alc.ParamsDefinitionSet):
    def __init__(self):
        self.params_definition_set =\
            [
                alc.ParamsDefinition(name='gamma', type='float', range='0,1', default_value='auto', description=''),
                alc.ParamsDefinition(name='nu', type='float', range='0,1', default_value='0.5', description=''),
                alc.ParamsDefinition(name='kernel', type='enum', range='linear,poly,rbf,sigmoid,precomputed', default_
                alc.ParamsDefinition(name='degree', type='int', range='', default_value='3', description=''),
            ]
```

繼承 algo_component 中的 ParamsDefinitionSet 類別



linear_regression.py

```python

class ParamsDefinitionSet(alc.ParamsDefinitionSet):
    def __init__(self):
        self.params_definition_set =\
            [
                alc.ParamsDefinition(name='fit_intercept', type='boolean', range='True,False', default_value='True'
                alc.ParamsDefinition(name='normalize', type='boolean', range='True,False', default_value='False', d
                alc.ParamsDefinition(name='copy_X', type='boolean', range='True,False', default_value='True', descr
                alc.ParamsDefinition(name='n_jobs', type='int', range='', default_value='1', description='')
            ]
```


scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html
Previous sklearn.svm.NuSVR
Next sklearn.svm.SVC
Up API Reference
scikit-learn v0.19.1
Other versions
Please cite us if you use the software.
sklearn.svm.OneClassSVM
Examples using sklearn.svm.OneClassSVM
«

sklearn.svm.OneClassSVM

class `sklearn.svm.` **`OneClassSVM`** (*kernel='rbf', degree=3, gamma='auto', coef0=0.0, tol=0.001, nu=0.5, shrinking=True, cache_size=200, verbose=False, max_iter=-1, random_state=None*) [source]

Unsupervised Outlier Detection.

Estimate the support of a high-dimensional distribution.

The implementation is based on libsvm.

Read more in the User Guide.

Parameters:

kernel : string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix.

nu : float, optional

An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Should be in the interval (0, 1]. By default 0.5 will be taken.

degree : int, optional (default=3)

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

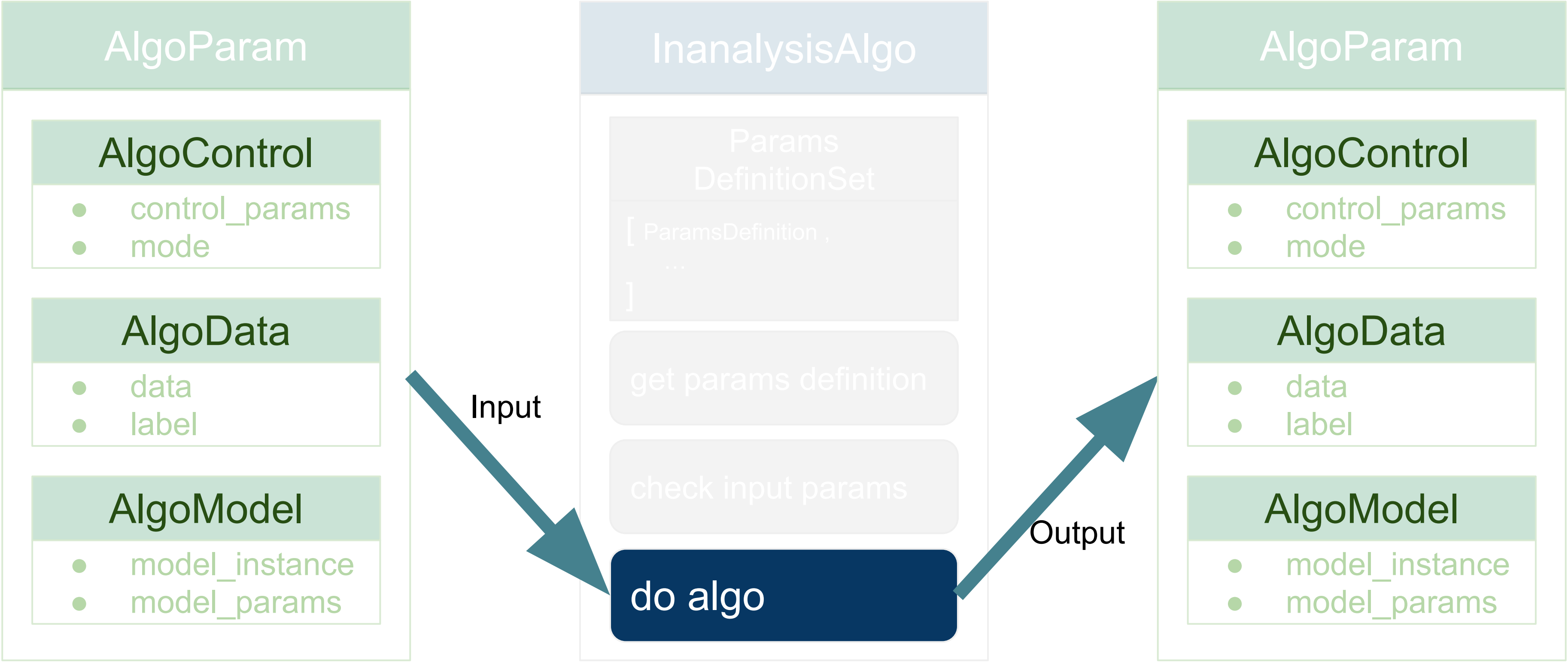
gamma : float, optional (default='auto')

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. If gamma is 'auto' then 1/n_features will be used instead.

ParamsDefinition中的type和range格式

type	range
‘int’	● ‘1,10’（Range is between 1 and 10） ● ‘’（Range is any integer number）
‘float’	● ‘0,1’（Range is between 0 and 1） ● ‘’（Range is any real number）
‘boolean’	● ‘True,False’
‘enum’	● ‘linear,poly,rbf,sigmoid,precomputed’（列出可選擇的字串）

演算法函式實作



AlgoParam

AlgoControl

- control_params
- mode

AlgoData

- data
- label

AlgoModel

- model_instance
- model_params

Input

InanalysisAlgo

Params DefinitionSet

[ParamsDefinition ,
…
]

get params definition

check input params

do algo

Output

AlgoParam

AlgoControl

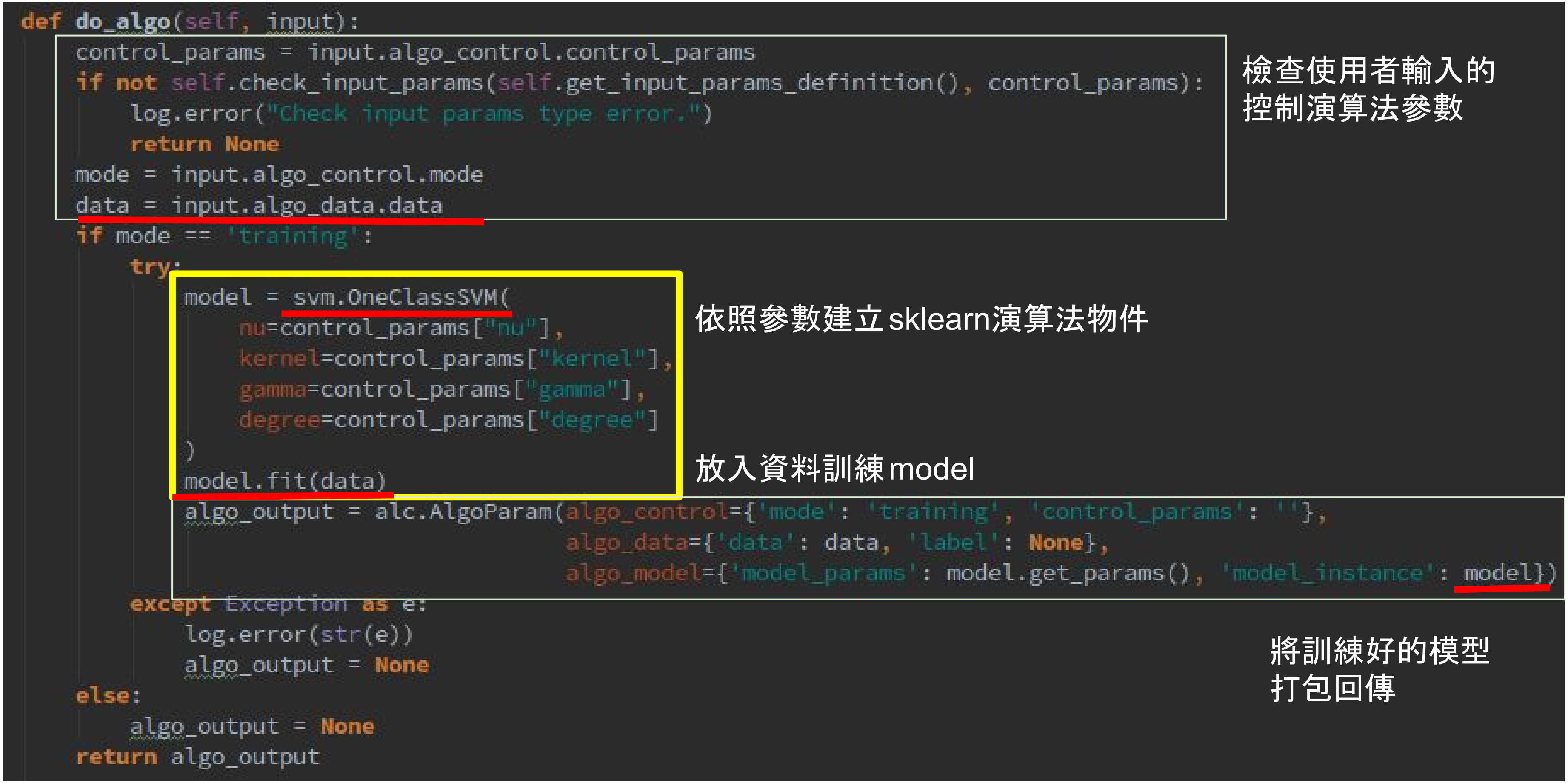
- control_params
- mode

AlgoData

- data
- label

AlgoModel

- model_instance
- model_params



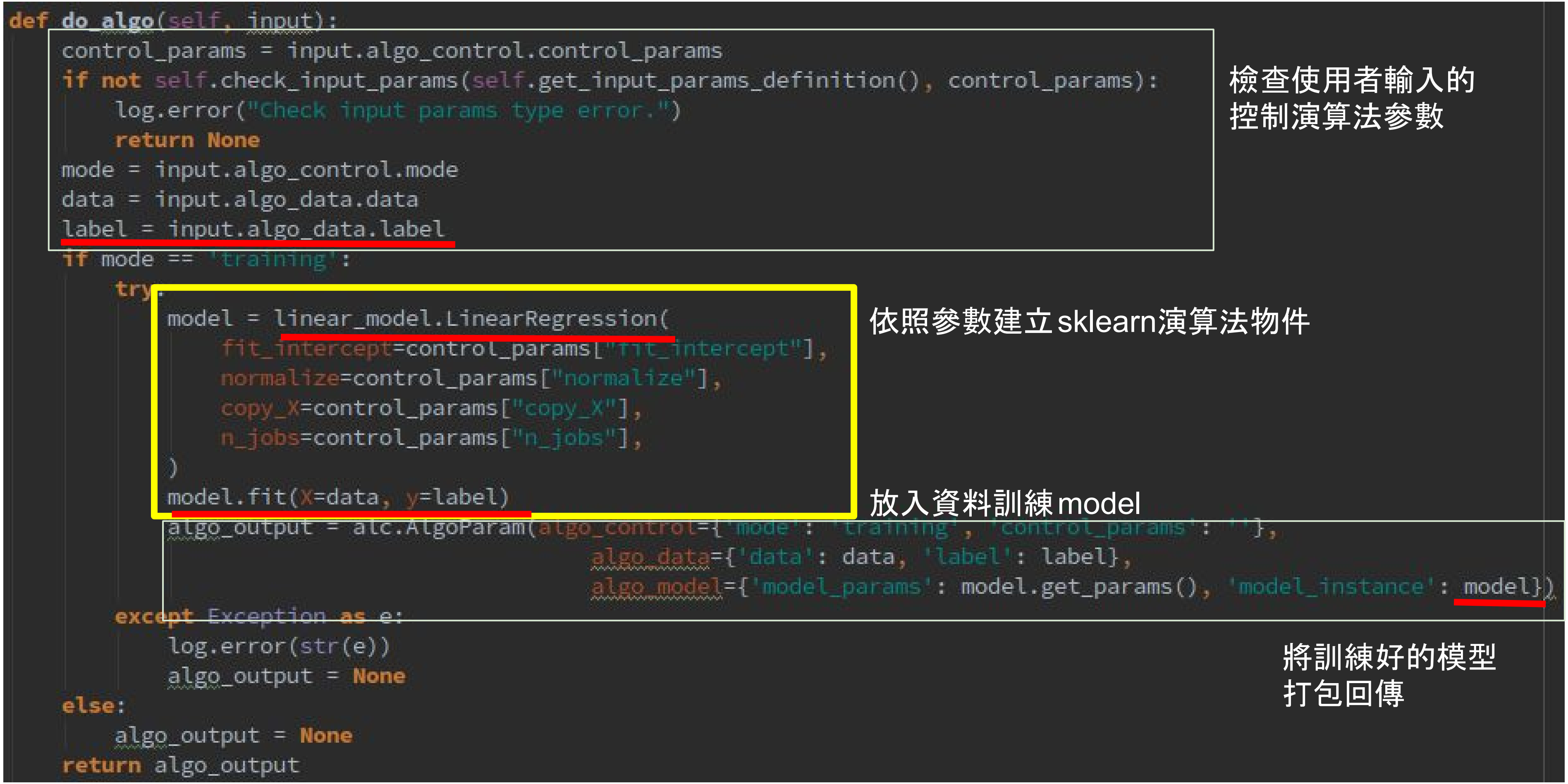
```
def do_algo(self, input):
    control_params = input.algo_control.control_params
    if not self.check_input_params(self.get_input_params_definition(), control_params):
        log.error("Check input params type error.")
        return None
    mode = input.algo_control.mode
    data = input.algo_data.data
    if mode == 'training':
        try:
            model = svm.OneClassSVM(
                nu=control_params["nu"],
                kernel=control_params["kernel"],
                gamma=control_params["gamma"],
                degree=control_params["degree"]
            )
            model.fit(data)
            algo_output = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': ''},
                                        algo_data={'data': data, 'label': None},
                                        algo_model={'model_params': model.get_params(), 'model_instance': model})
        except Exception as e:
            log.error(str(e))
            algo_output = None
    else:
        algo_output = None
    return algo_output
```

檢查使用者輸入的
控制演算法參數

依照參數建立sklearn演算法物件

放入資料訓練model

將訓練好的模型
打包回傳



```python
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.LinearRegression(
                fit_intercept=control_params["fit_intercept"],
                normalize=control_params["normalize"],
                copy_X=control_params["copy_X"],
                n_jobs=control_params["n_jobs"],
            )
            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:
            log.error(str(e))
            algo_output = None
    else:
        algo_output = None
    return algo_output
```

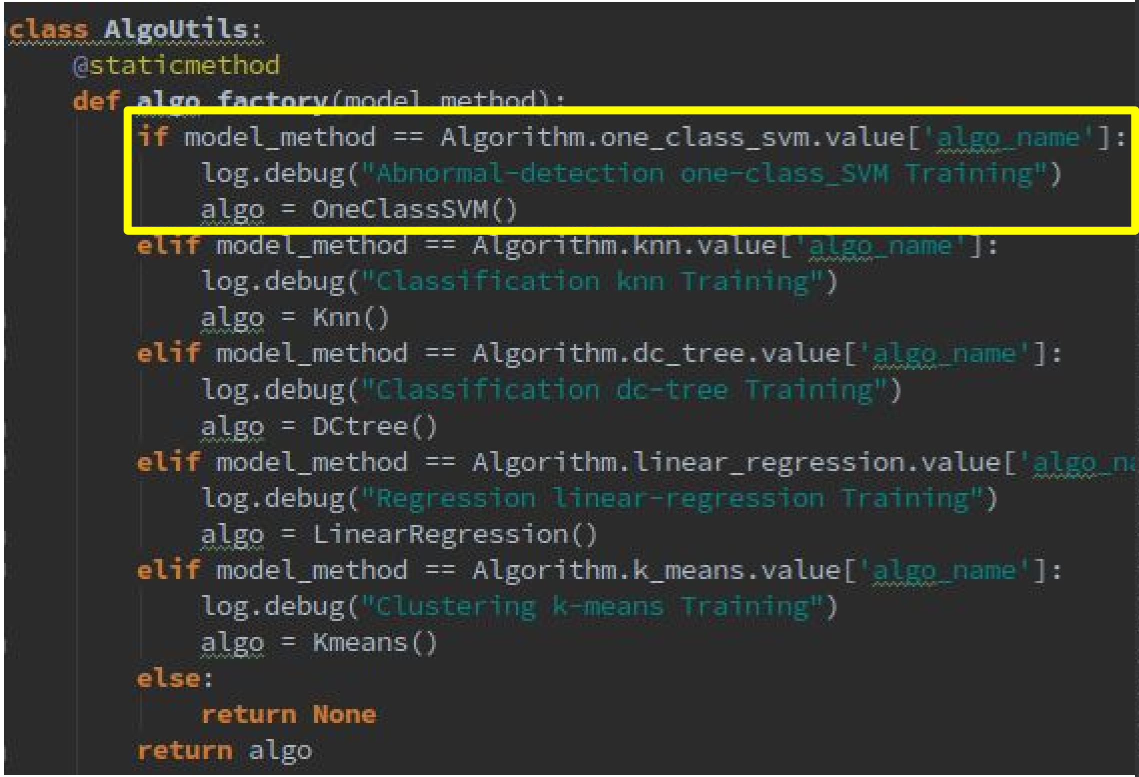
檢查使用者輸入的
控制演算法參數

依照參數建立sklearn演算法物件

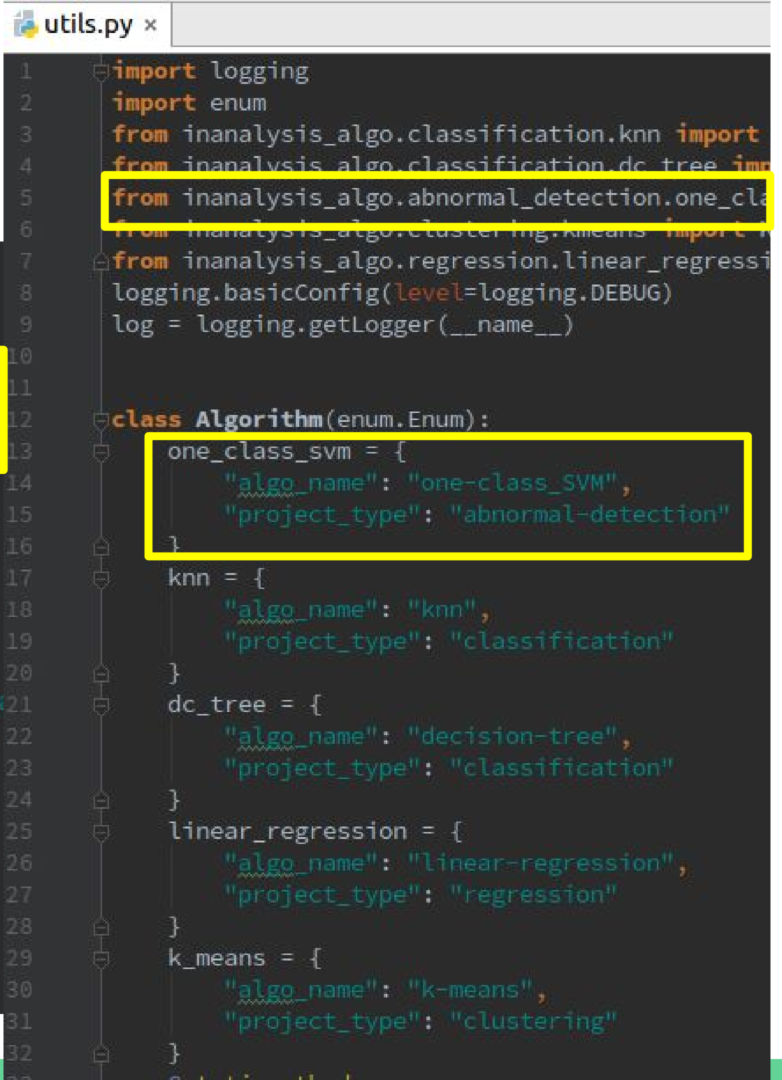
放入資料訓練model

將訓練好的模型
打包回傳

建立演算法物件工具



```python
class AlgoUtils:
    @staticmethod
    def algo_factory(model_method):
        if model_method == Algorithm.one_class_svm.value['algo_name']:
            log.debug("Abnormal-detection one-class_SVM Training")
            algo = OneClassSVM()
        elif model_method == Algorithm.knn.value['algo_name']:
            log.debug("Classification knn Training")
            algo = Knn()
        elif model_method == Algorithm.dc_tree.value['algo_name']:
            log.debug("Classification dc-tree Training")
            algo = DCtree()
        elif model_method == Algorithm.linear_regression.value['algo_na
            log.debug("Regression linear-regression Training")
            algo = LinearRegression()
        elif model_method == Algorithm.k_means.value['algo_name']:
            log.debug("Clustering k-means Training")
            algo = Kmeans()
        else:
            return None
        return algo
```



utils.py ×

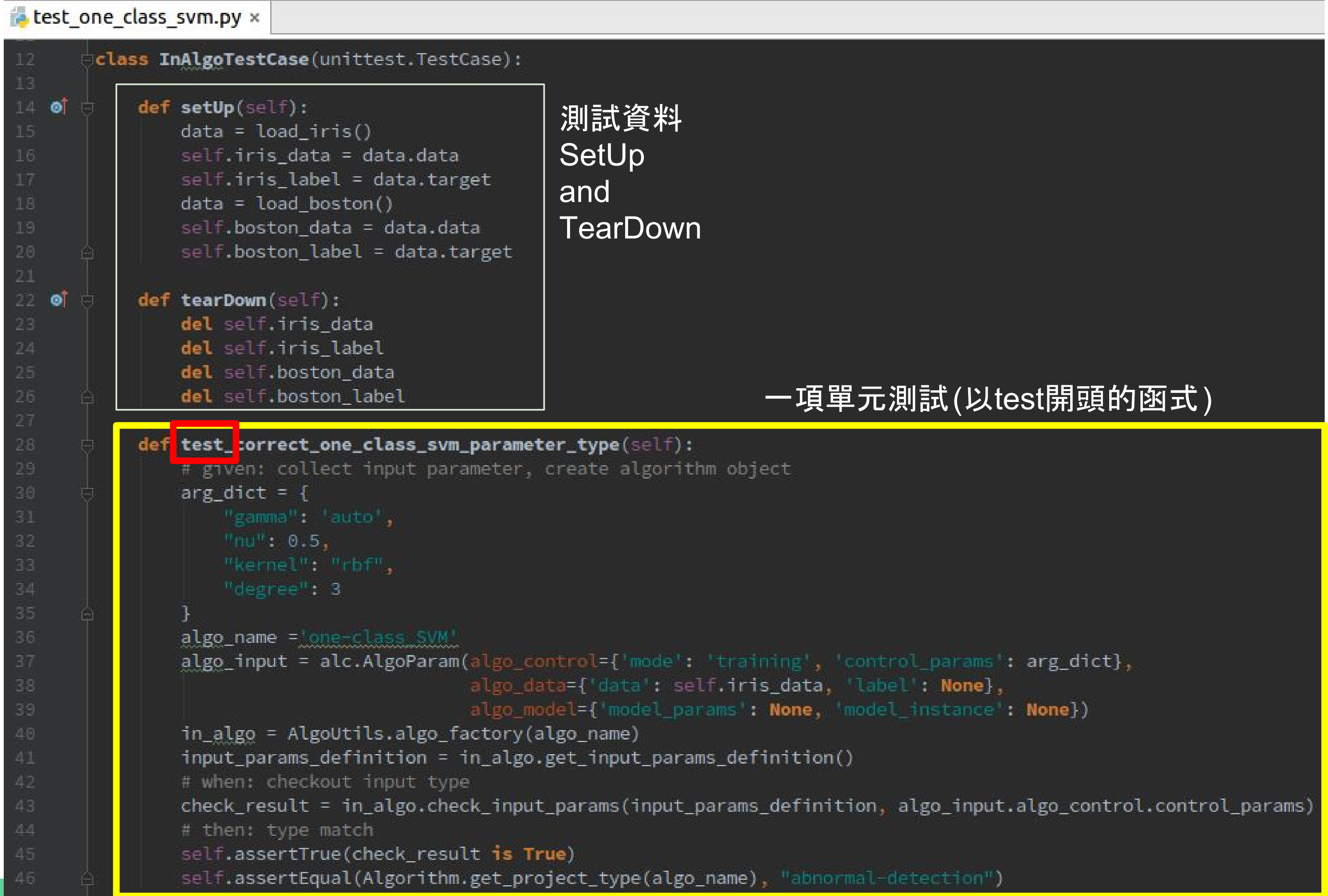
```python
import logging
import enum
from inanalysis_algo.classification.knn import
from inanalysis_algo.classification.dc_tree imp
from inanalysis_algo.abnormal_detection.one_cla
from inanalysis_algo.clustering.kmeans import K
from inanalysis_algo.regression.linear_regressi
logging.basicConfig(level=logging.DEBUG)
log = logging.getLogger(__name__)


class Algorithm(enum.Enum):
    one_class_svm = {
        "algo_name": "one-class_SVM",
        "project_type": "abnormal-detection"
    }
    knn = {
        "algo_name": "knn",
        "project_type": "classification"
    }
    dc_tree = {
        "algo_name": "decision-tree",
        "project_type": "classification"
    }
    linear_regression = {
        "algo_name": "linear-regression",
        "project_type": "regression"
    }
    k_means = {
        "algo_name": "k-means",
        "project_type": "clustering"
    }
```

2. 演算法測試實作

單元測試 Unittest

- unittest 有時亦稱為 “PyUnit", 是 JUnit 的 Python 語言實現, JUnit是個單元測試(Unit test)框架, 單元測試指的是測試一個工作單元(a unit of work)的行為。
- 就軟體測試而言, 單元測試通常指的是測試某個函式(或方法), 你**給予該函式某些輸入, 預期該函式會產生某種輸出**, 例如傳回預期的值、產生預期的檔案、新增預期的資料等。
- Given-When-Then
  - Given - 給予該函式某些輸入
  - When - 執行該函式
  - Then - 預期該函式會 產生某種輸出

reference : http://www.codedata.com.tw/python/python-tutorial-the-6th-class-1-unittest/

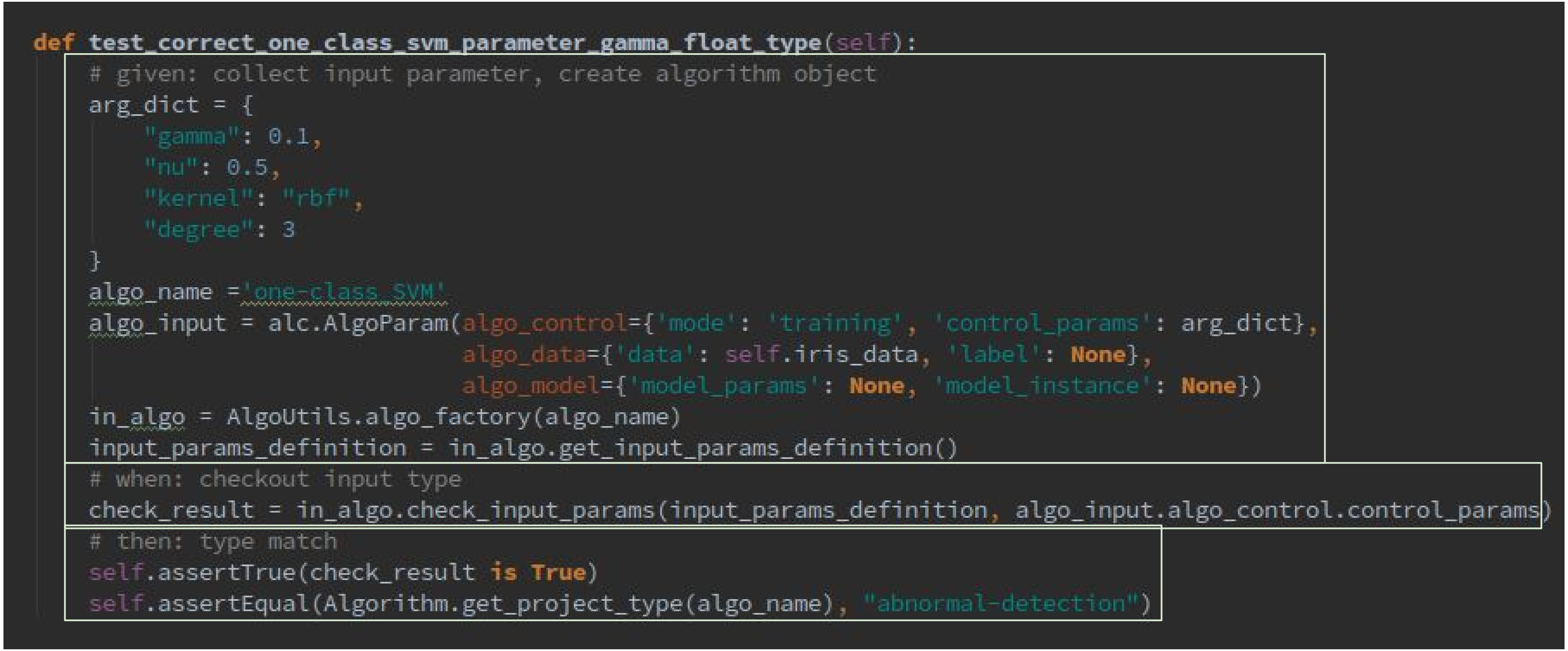


```python
class InAlgoTestCase(unittest.TestCase):

    def setUp(self):
        data = load_iris()
        self.iris_data = data.data
        self.iris_label = data.target
        data = load_boston()
        self.boston_data = data.data
        self.boston_label = data.target

    def tearDown(self):
        del self.iris_data
        del self.iris_label
        del self.boston_data
        del self.boston_label

    def test_correct_one_class_svm_parameter_type(self):
        # given: collect input parameter, create algorithm object
        arg_dict = {
            "gamma": 'auto',
            "nu": 0.5,
            "kernel": "rbf",
            "degree": 3
        }
        algo_name ='one-class SVM'
        algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
                                   algo_data={'data': self.iris_data, 'label': None},
                                   algo_model={'model_params': None, 'model_instance': None})
        in_algo = AlgoUtils.algo_factory(algo_name)
        input_params_definition = in_algo.get_input_params_definition()
        # when: checkout input type
        check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
        # then: type match
        self.assertTrue(check_result is True)
        self.assertEqual(Algorithm.get_project_type(algo_name), "abnormal-detection")
```

測試資料
SetUp
and
TearDown

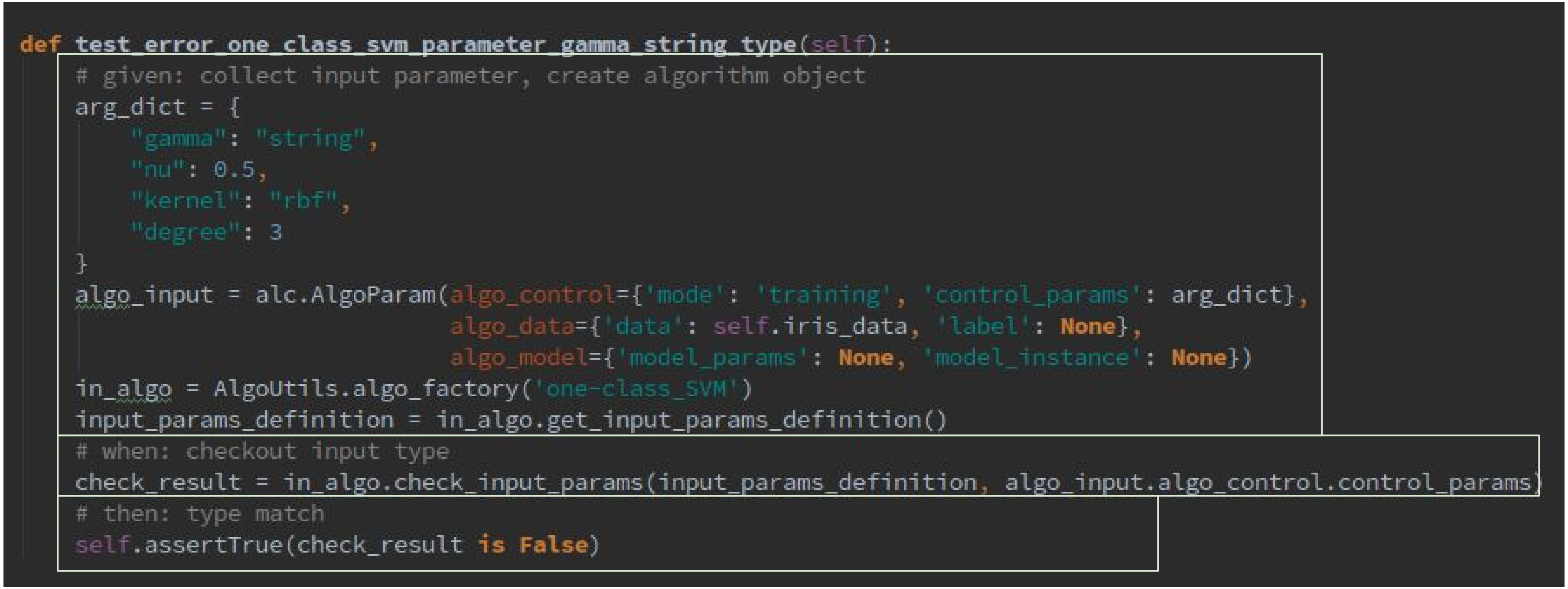
一項單元測試(以test開頭的函式)

Happy Face Test :)



```
def test_correct_one_class_svm_parameter_gamma_float_type(self):
    # given: collect input parameter, create algorithm object
    arg_dict = {
        "gamma": 0.1,
        "nu": 0.5,
        "kernel": "rbf",
        "degree": 3
    }
    algo_name ='one-class SVM'
    algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
                               algo_data={'data': self.iris_data, 'label': None},
                               algo_model={'model_params': None, 'model_instance': None})
    in_algo = AlgoUtils.algo_factory(algo_name)
    input_params_definition = in_algo.get_input_params_definition()
    # when: checkout input type
    check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
    # then: type match
    self.assertTrue(check_result is True)
    self.assertEqual(Algorithm.get_project_type(algo_name), "abnormal-detection")
```

Sad Face Test :(

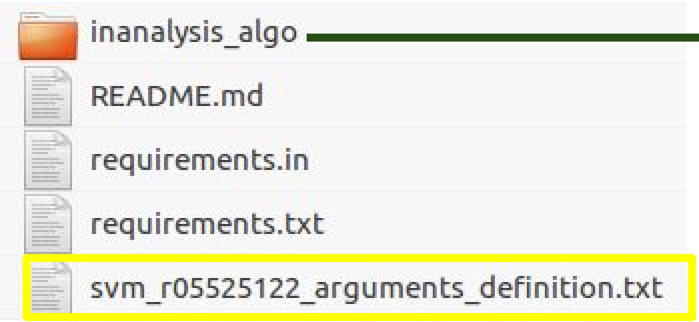


```
def test_error_one_class_svm_parameter_gamma_string_type(self):
    # given: collect input parameter, create algorithm object
    arg_dict = {
        "gamma": "string",
        "nu": 0.5,
        "kernel": "rbf",
        "degree": 3
    }
    algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
                               algo_data={'data': self.iris_data, 'label': None},
                               algo_model={'model_params': None, 'model_instance': None})
    in_algo = AlgoUtils.algo_factory('one-class_SVM')
    input_params_definition = in_algo.get_input_params_definition()
    # when: checkout input type
    check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
    # then: type match
    self.assertTrue(check_result is False)
```

演算法模組作業評分說明

- Deadline：
- 計分方法：
  - 演算法模組實作 (10 points)
    - 是否正確實作演算法模組
  - 演算法測試實作 (15 points)
    - 是否有測試到各種情況（正向測試, 負向測試都要涵蓋）
  - 程式編寫可讀性 (5 points)
    - 函式與變數命名是否明確,是否有註解......等

需要修改的檔案



inanalysis_algo

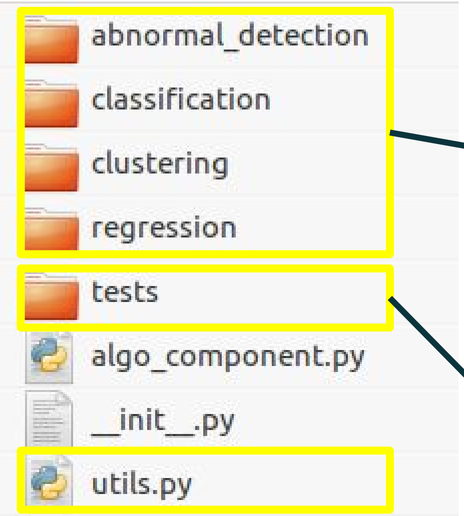
README.md

requirements.in

requirements.txt

svm_r05525122_arguments_definition.txt

演算法參數定義，
演算法sklearn網站連結



abnormal_detection

classification

clustering

regression

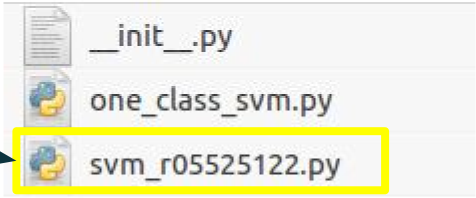
tests

algo_component.py

__init__.py

utils.py

加上演算法選擇工具選項

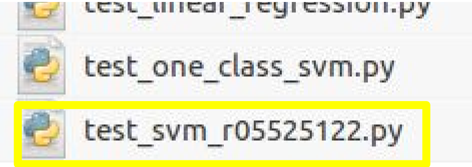


__init__.py

one_class_svm.py

svm_r05525122.py

演算法實作



test_linear_regression.py

test_one_class_svm.py

test_svm_r05525122.py

演算法測試