

A top-down view of a desk with various items: a small potted plant in the top left, a black wallet with a gold cross in the top center, a pair of black-rimmed glasses in the top right, a hand holding a black pencil on the right side, and a stack of papers and envelopes in the center. A laptop is partially visible on the left.

Final Project

Lasso regression

A person is walking away from the camera on a wooden floor. They are wearing a blue and white plaid shirt, dark blue pants, and grey sneakers. They are carrying a brown leather backpack. The background is slightly blurred, showing a wooden chair and a wall.

hello!

**I AM  
BERRIE  
CHEN**

Name: 陳昱芝

ID: r06525089

抽籤編號: 10[[演算法文件](#)]

go40631@gmail.com



# Outline

Algorithm Introduction



Code Review



Model Review



Live Demo



Conclusion



Reference

A dimly lit, warm-toned photograph of a wooden table in what appears to be a cafe or meeting space. In the background, a large, dark brown leather bag sits on the table. In the foreground, a clear glass of water is partially filled. To the right, a hand is visible, resting near a smartphone and a tablet. Another smartphone lies on the table to the left of the glass. A white coffee cup on a saucer is partially visible in the bottom left corner. The background is a brick wall. The overall atmosphere is quiet and professional.

# 1. Algorithm Introduction



# Algorithm Introduction

LASSO (least absolute shrinkage and selection operator):

- ⊙ A regularization regression method.
- ⊙ Perform both **feature selection** and **regularization**.

$$\min_w \frac{1}{2n_{\text{samples}}} \|Xw - y\|_2^2 + \alpha \|w\|_1$$

↓  
最小平方和

↓  
正規化  
(regularizer)

- ⊙ Trained with L1 prior as regularizer.

A photograph of a person's hands writing in a small notebook on a wooden table. A black leather bag is in the background. On the table are a glass of water, a smartphone, and a tablet. A coffee cup is also visible. The scene is dimly lit with a brick wall in the background.

## 2. Code Review

algo\_component.py x

```
1 import ...
4 logging.basicConfig(level=logging.DEBUG)
5 log = logging.getLogger(__name__)
6
7
8 class ParamsDefinition:
9     def __init__(self, name, type, range, default_value, description):
10         self.name = name
11         self.type = type
12         self.range = range
13         self.default_value = default_value
14         self.description = description
15
16     def get_params_definition(self):
17         return self.__dict__
18
19
20 class ParamsDefinitionSet:
21     def __init__(self):
22         self.params_definition_set = []
23         raise NotImplementedError
24
25     def get_params_definition_set(self):
26         definition_set_json_list = []
27         for params_object in self.params_definition_set:
28             definition_set_json_list.append(params_object.get_params_definition())
29         return definition_set_json_list
```

→ 在各演算法  
子類別中實作

Code Review



lasso\_regression\_r06525089.py

```
1 from sklearn import linear_model
2 import inanalysis_algo.algo_component as alc
3 import logging
4 logging.basicConfig(level=logging.DEBUG)
5 log = logging.getLogger(__name__)
6
7
8 class ParamsDefinitionSet(alc.ParamsDefinitionSet):
9     def __init__(self):
10         self.params_definition_set = \
11             {
12                 alc.ParamsDefinition(name='alpha', type='float', range='0,1', default_value='1.0', description=''),
13                 alc.ParamsDefinition(name='fit_intercept', type='boolean', range='True,False', default_value='True', description=''),
14                 alc.ParamsDefinition(name='normalize', type='boolean', range='True,False', default_value='False', description=''),
15                 alc.ParamsDefinition(name='copy_X', type='boolean', range='True,False', default_value='True', description='')
16             }
17
```

繼承 algo\_component 中的  
ParamsDefinitionSet 類別

Code Review



```

lasso_regression_r06525089.py x
1  from sklearn import linear_model
2  import inanalysis_algo.algo_component as alc
3  import logging
4  logging.basicConfig(level=logging.DEBUG)
5  log = logging.getLogger(__name__)
6
7
8  class ParamsDefinitionSet(alc.ParamsDefinitionSet):...
17
18
19  class LassoRegression(alc.InanalysisAlgo):
20  def __init__(self):
21  self.input_params_definition = ParamsDefinitionSet()
22
23  def get_input_params_definition(self):
24  return self.input_params_definition.get_params_definition_json_list()
25
26  def do_algo(self, input):
27  control_params = input.algo_control.control_params
28  if not self.check_input_params(self.get_input_params_definition(), control_params):
29  log.error("Check input params type error.")
30  return None
31
32  mode = input.algo_control.mode
33  data = input.algo_data.data
34  label = input.algo_data.label
35  if mode == 'training':
36  try:
37  model = linear_model.Lasso(
38  alpha=control_params["alpha"],
39  fit_intercept=control_params["fit_intercept"],
40  normalize=control_params["normalize"],
41  copy_X=control_params["copy_X"],

```

依照輸入參數建立  
sklearn.linear\_model.Lasso

物件

} 較為重要的參數  
作為可調整的參數  
讓使用者輸入

Code Review

```
26 def do_algo(self, input):
27     control_params = input.algo_control.control_params
28     if not self.check_input_params(self.get_input_params_definition(), control_params):
29         log.error("Check input params type error.")
30         return None
31     mode = input.algo_control.mode
32     data = input.algo_data.data
33     label = input.algo_data.label
34     if mode == 'training':
35         try:
36             model = linear_model.Lasso(
37                 alpha=control_params["alpha"],
38                 fit_intercept=control_params["fit_intercept"],
39                 normalize=control_params["normalize"],
40                 copy_X=control_params["copy_X"],
41             )
42             model.fit(X=data, y=label)
43             algo_output = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': ''},
44                                         algo_data={'data': data, 'label': label},
45                                         algo_model={'model_params': model.get_params(), 'model_instance': model})
```

放入資料訓練模型

輸出訓練好的模型

Code Review

```
22 def test_correct_lasso_regression_parameter_type(self):
23     # given: collect input parameter, create algorithm object
24     arg_dict = {
25         "alpha": 1.0,
26         "fit_intercept": True,
27         "normalize": False,
28         "copy_X": True
29     }
30     algo_name = 'lasso'
31     algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
32                                algo_data={'data': self.boston_data, 'label': self.boston_label},
33                                algo_model={'model_params': None, 'model_instance': None})
34     in_algo = AlgoUtils.algo_factory(algo_name)
35     input_params_definition = in_algo.get_input_params_definition()
36     # when: checkout input type
37     check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
38     # then: type match
39     self.assertTrue(check_result is True)
40     self.assertEqual(Algorithm.get_project_type(algo_name), "regression")
```

參數測試 Happy Path Test

Code Review



```

42 def test_error_lasso_regression_parameter_alpha_string_type(self):
43     # given: error input parameter "alpha" needs to be type(float)
44     arg_dict = {
45         "alpha": 'string',
46         "fit_intercept": True,
47         "normalize": False,
48         "copy_X": True
49     }
50     algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
51                               algo_data={'data': self.boston_data, 'label': self.boston_label},
52                               algo_model={'model_params': None, 'model_instance': None})
53     in_algo = AlgoUtils.algo_factory('lasso')
54     input_params_definition = in_algo.get_input_params_definition()
55
56     # when: checkout input type
57     check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
58     # then: type match
59     self.assertTrue(check_result is False)

```

參數測試 Sad Path Test

Code Review

```

61 def test_error_lasso_regression_parameter_fit_intercept_string_type(self):
62     # given: error input parameter "fit_intercept" needs to be type(boolean)
63     arg_dict = {
64         "alpha": 1.0,
65         "fit_intercept": 'string'
66         "normalize": False,
67         "copy_X": True
68     }
69     algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
70                               algo_data={'data': self.boston_data, 'label': self.boston_label},
71                               algo_model={'model_params': None, 'model_instance': None})
72     in_algo = AlgoUtils.algo_factory('lasso')
73     input_params_definition = in_algo.get_input_params_definition()
74
75     # when: checkout input type
76     check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
77     # then: type match
78     self.assertTrue(check_result is False)

```

參數測試 Sad Path Test

Code Review

```

99 def test_error_lasso_regression_parameter_fit_intercept_float_type(self):
100     # given: error input parameter "fit_intercept" needs to be type(boolean)
101     arg_dict = {
102         "alpha": 1.0,
103         "fit_intercept": 1.0,
104         "normalize": False,
105         "copy_X": True
106     }
107     algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
108                               algo_data={'data': self.boston_data, 'label': self.boston_label},
109                               algo_model={'model_params': None, 'model_instance': None})
110     in_algo = AlgoUtils.algo_factory('lasso')
111     input_params_definition = in_algo.get_input_params_definition()
112
113     # when: checkout input type
114     check_result = in_algo.check_input_params(input_params_definition, algo_input.algo_control.control_params)
115     # then: type match
116     self.assertTrue(check_result is False)

```

參數測試 Sad Path Test

Code Review



```

118 def test_correct_lasso_regression_do_algo(self):
119     # given: collect input parameter, create algorithm object
120     arg_dict = {
121         "alpha": 1.0,
122         "fit_intercept": True,
123         "normalize": False,
124         "copy_X": True
125     }
126     algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
127                               algo_data={'data': self.boston_data, 'label': self.boston_label},
128                               algo_model={'model_params': None, 'model_instance': None})
129     in_algo = AlgoUtils.algo_factory('lasso')
130     log.debug(algo_input)
131     # when: do decision tree algorithm
132     algo_output = in_algo.do_algo(algo_input)
133
134     # then:
135     self.assertTrue(algo_output is not None)
136     self.assertTrue(algo_output.algo_model.model_instance is not None)
137     self.assertTrue(algo_output.algo_model.model_instance.predict(X=self.boston_data) is not None)
138     #self.assertTrue(algo_output.algo_model.model_instance.score(X=self.boston_data, y=self.boston_label) is float)

```

演算法測試 Happy Path Test

Sklearn官方文件  
Lasso之Methods

Methods
<code>fit(X, y[, check_input])</code>
<code>get_params([deep])</code>
<code>path(X, y[, l1_ratio, eps, n_alphas, ...])</code>
<code>predict(X)</code>
<code>score(X, y[, sample_weight])</code>
<code>set_params(**params)</code>

模型訓練成功

用訓練好的模型作預測

Code Review

A photograph of a person's hands writing in a small notebook on a wooden table. A large black leather bag is in the background. On the table are a glass of water, a smartphone, and a tablet. A coffee cup is also visible in the lower left. The scene is dimly lit with a brick wall in the background.

### 3. Model Review

```

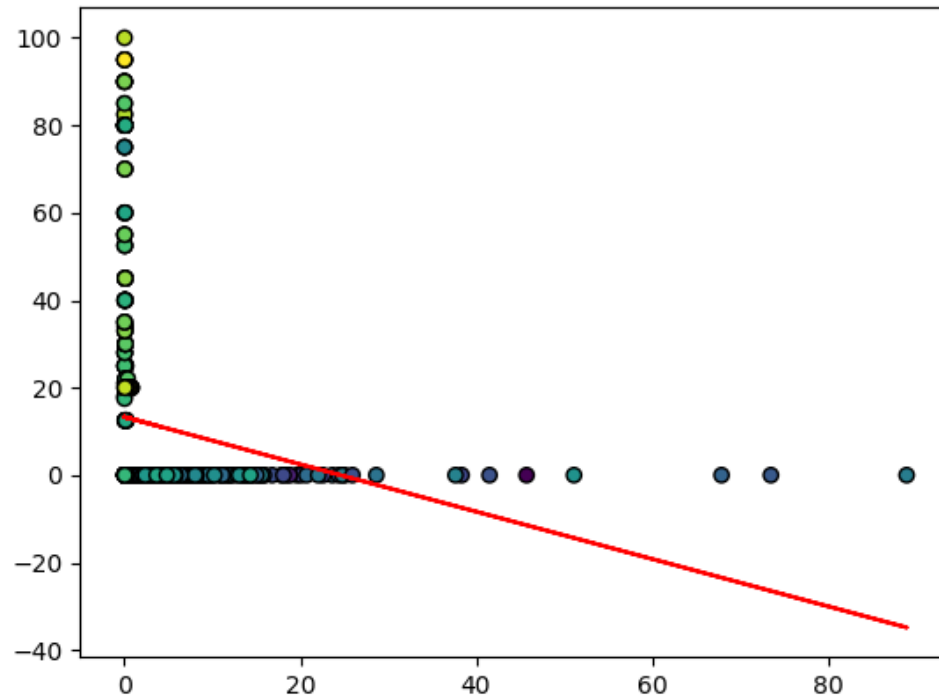
1  import matplotlib.pyplot as plt
2  import inanalysis_algo.algo_component as alc
3  from inanalysis_algo.utils import AlgoUtils
4  from sklearn.datasets import load_boston
5  import pandas as pd
6  import numpy as np
7
8
9  def lasso_model_preview(data, predict_result, x_axis_name, y_axis_name):
10     plt.title("Boston model plot")
11     plt.xlabel(x_axis_name)
12     plt.ylabel(y_axis_name)
13     fig, ax = plt.subplots()
14     fit = np.polyfit(data[x_axis_name], data[y_axis_name], deg=1)
15     ax.plot(data[x_axis_name], fit[0] * data[x_axis_name] + fit[1], color='red')
16     ax.scatter(data[x_axis_name], data[y_axis_name], c=predict_result, edgecolor='black', linewidth='1')
17     plt.savefig(x_axis_name + " _ " + y_axis_name + ".png")
18     plt.close('all')

```

Regression line

Model Review





Model Review

A dimly lit wooden table with a leather bag, a glass of water, a smartphone, a tablet, and hands writing on a notepad.

## 4. Live Demo

<http://ntuesoe.com:8008/>

A photograph of a person's hands writing in a small notebook on a wooden table. A large black leather bag is in the background. On the table are a glass of water, a smartphone, and a tablet. A coffee cup is partially visible in the bottom left. The scene is dimly lit with a brick wall in the background.

## 5. Conclusion





# What I have **learned**:

- ◎ 運用套件的method
- ◎ 寫單元測試(Unit Test)
- ◎ survey使用工具背後的相關技術
- ◎ 容易又快速的開發演算法程式

A photograph of a person's hands writing in a small notebook on a wooden table. A large black leather bag is in the background. On the table are a glass of water, a smartphone, and a tablet. A coffee cup is partially visible in the lower left. The scene is dimly lit with a brick wall in the background.

## 6. Reference

# Reference

[http://scikit-learn.org/stable/modules/linear\\_model.html#lasso](http://scikit-learn.org/stable/modules/linear_model.html#lasso)

[https://dotblogs.com.tw/dash\\_analysis/2017/11/03/161734](https://dotblogs.com.tw/dash_analysis/2017/11/03/161734)





A person is walking away from the camera on a wooden floor. They are wearing a red and blue plaid shirt, dark pants, and sneakers. They are carrying a brown backpack. The background is slightly blurred, showing a wooden chair and a wooden staircase.

thanks!

**ANY**  
**QUESTIONS?**