Python与机器学习

——机器学习算法

华算科技　黄老师

2022年2月22日

华算科技
HUASUAN TECHNOLOGY

目录

1. 算法简介
2. 回归算法
3. 双金属吸附中的回归
4. 非线性回归求速率常数
5. 模型评价
6. 决策树分类算法
7. 支持向量机

目录

1. 算法简介

2. 回归算法

3. 双金属吸附中的回归

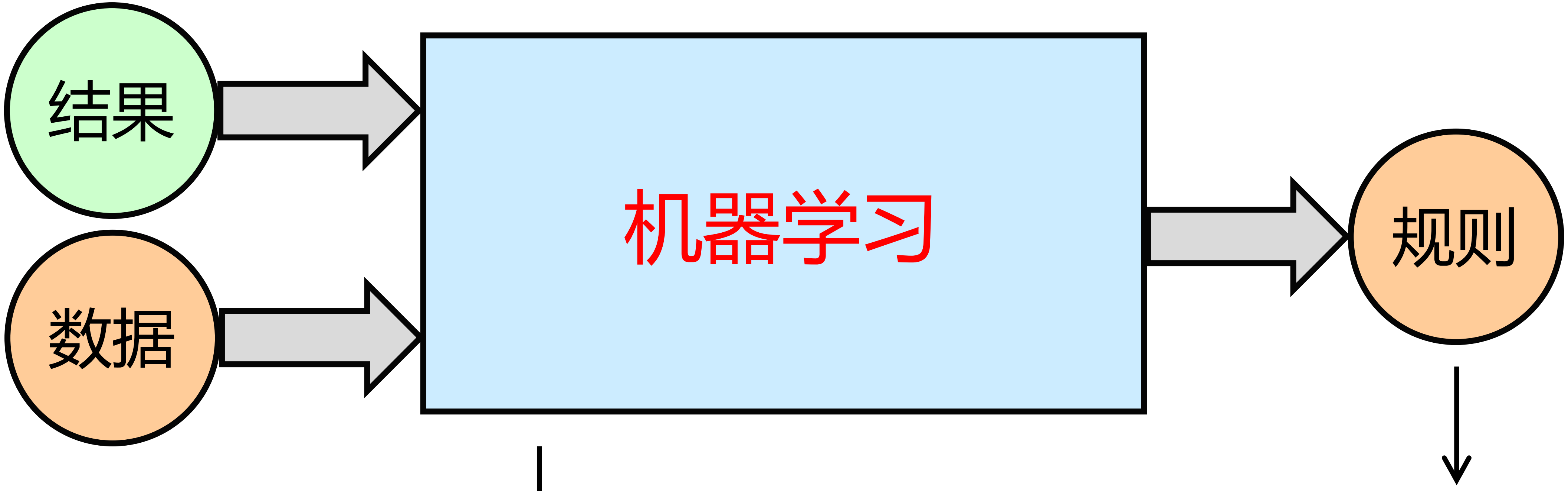
4. 非线性回归求速率常数

5. 模型评价

6. 决策树分类算法

7. 支持向量机

机器学习算法

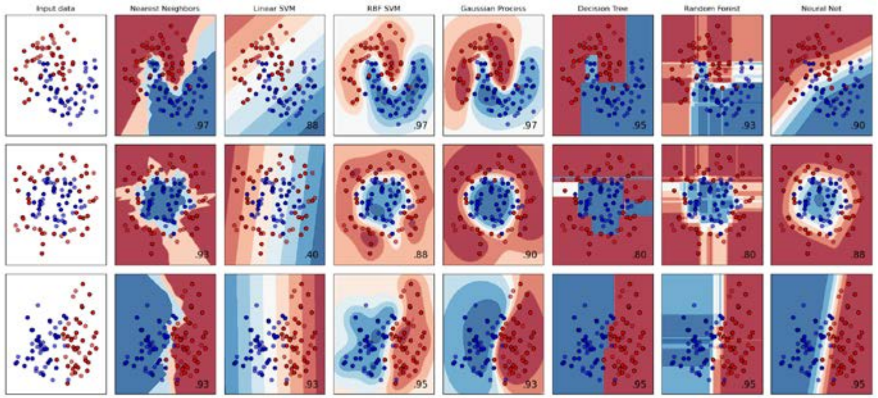


$$f(x_1, x_2, x_3, \cdots, x_n) = y$$

寻找最好的映射关系 f 的过程

算法

机器学习算法

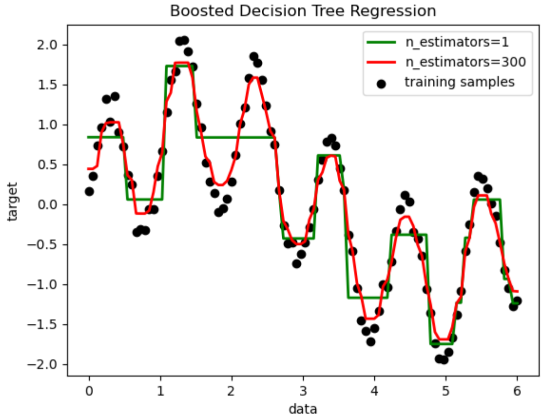


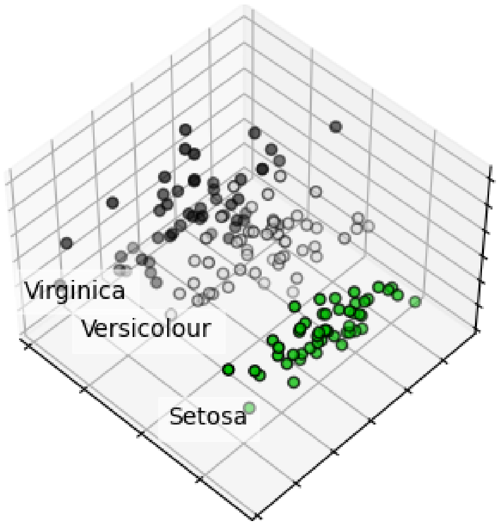
回归算法

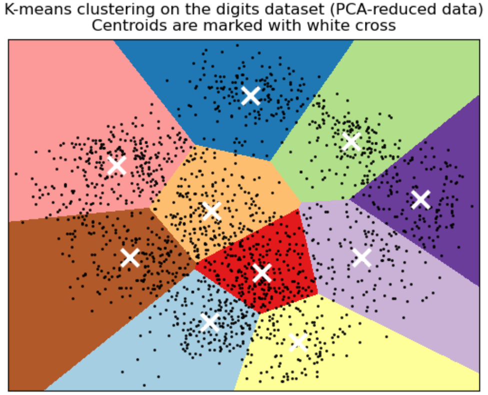
分类算法

聚类算法

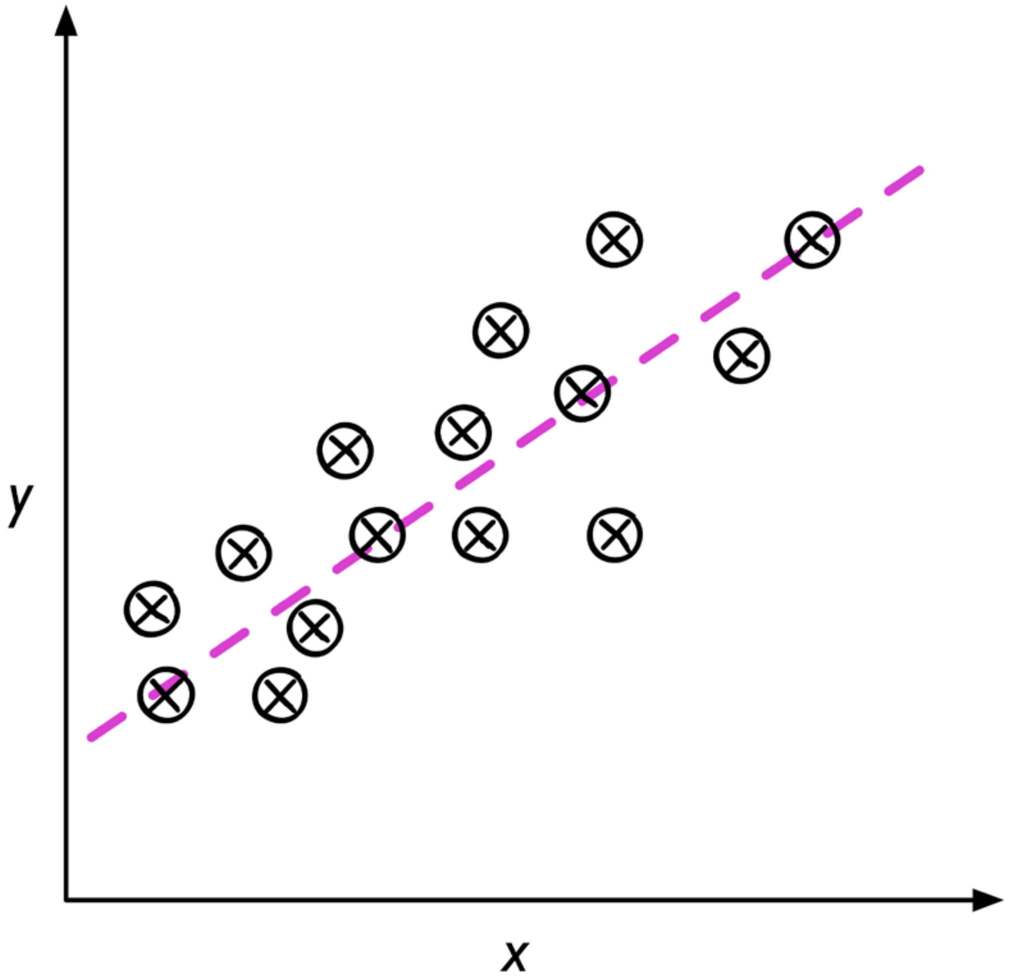
维度约减



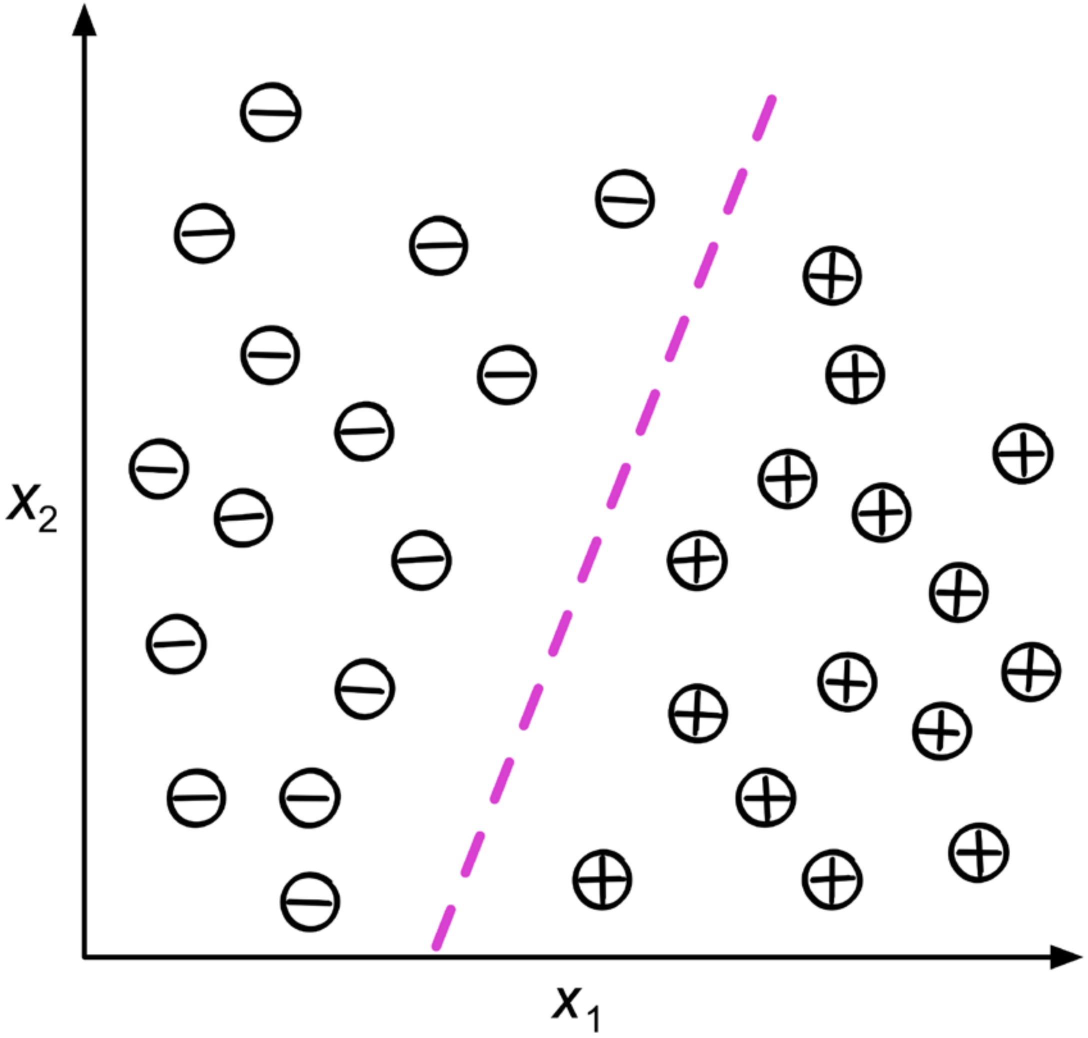




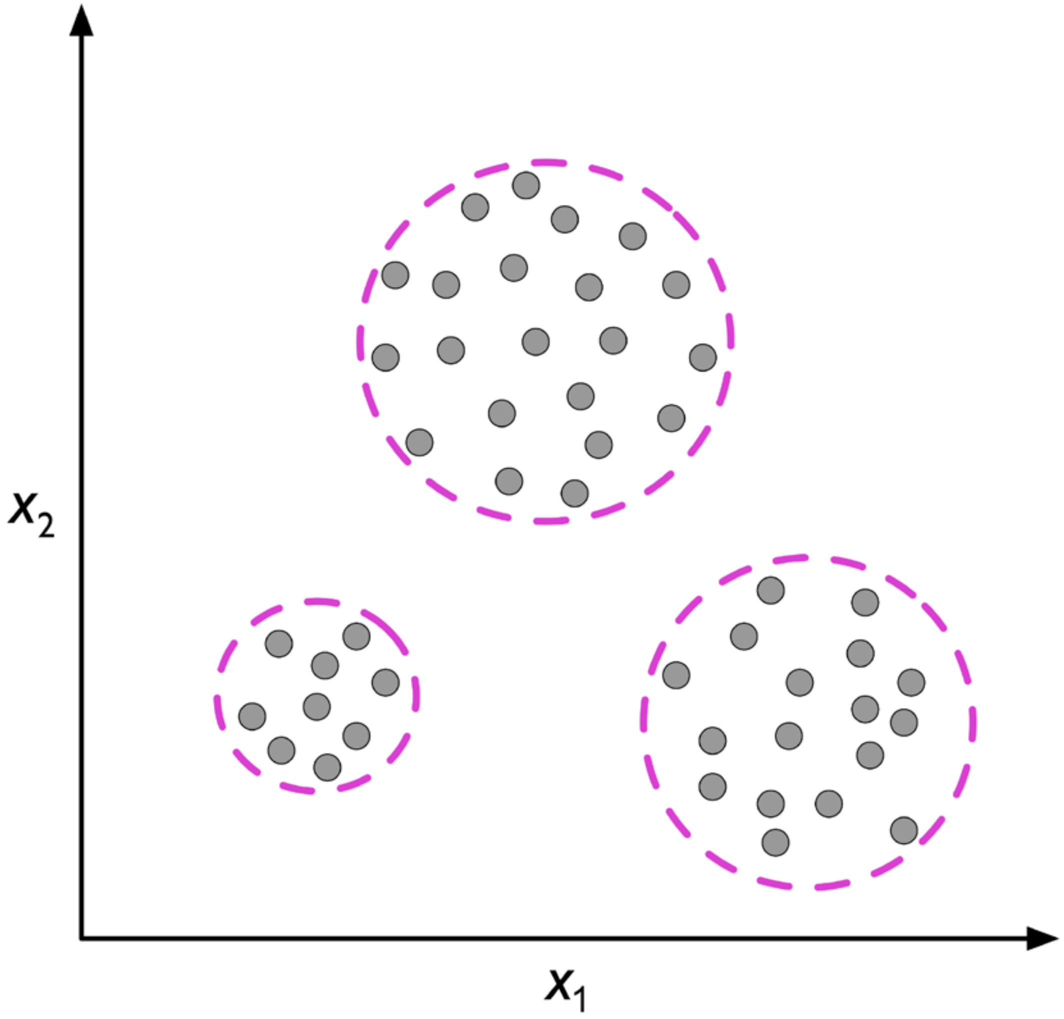
回归算法



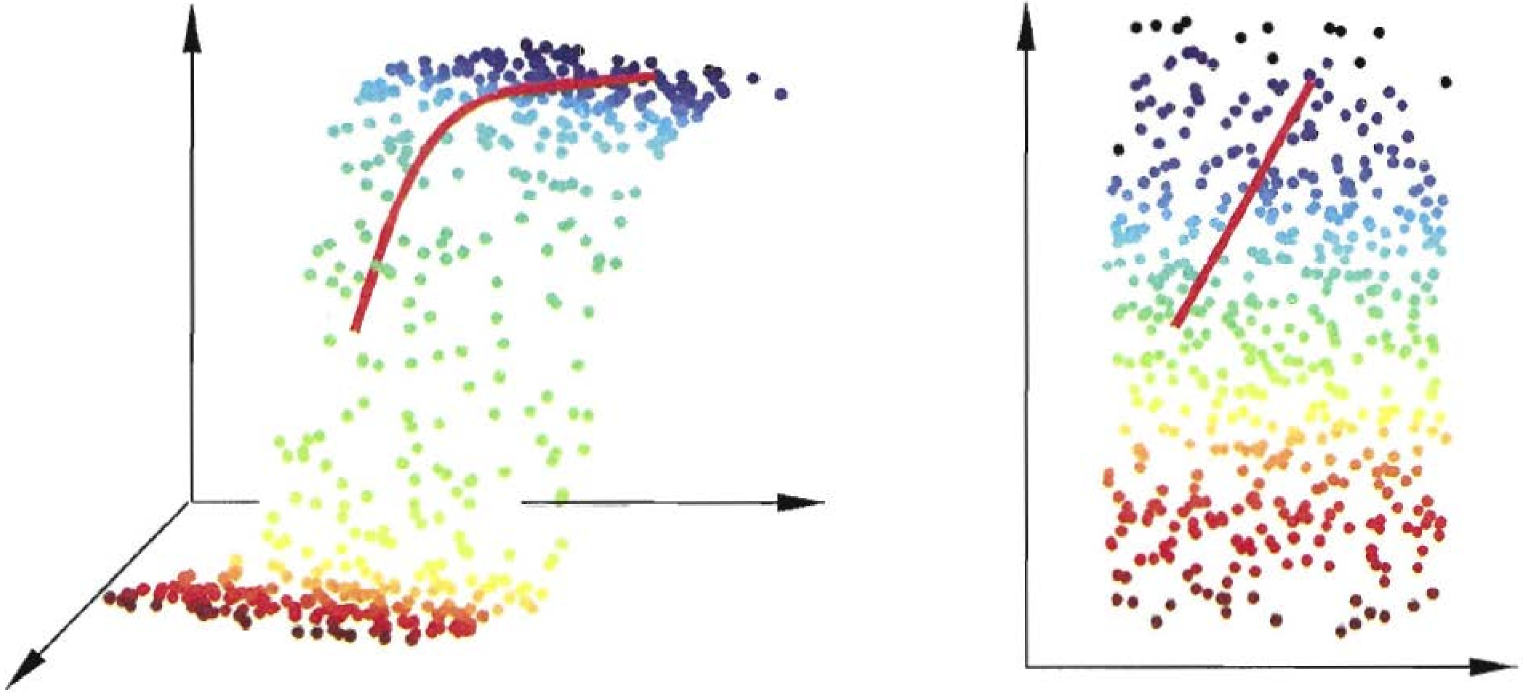
分类算法



聚类算法



维度约减



目录

1. 算法简介

2. 回归算法

3. 双金属吸附中的回归

4. 非线性回归求速率常数

5. 模型评价

6. 决策树分类算法

7. 支持向量机

回归算法

寻找目标值（通常是连续值）与输入值之间关系的算法。

线性回归
Linear Regression

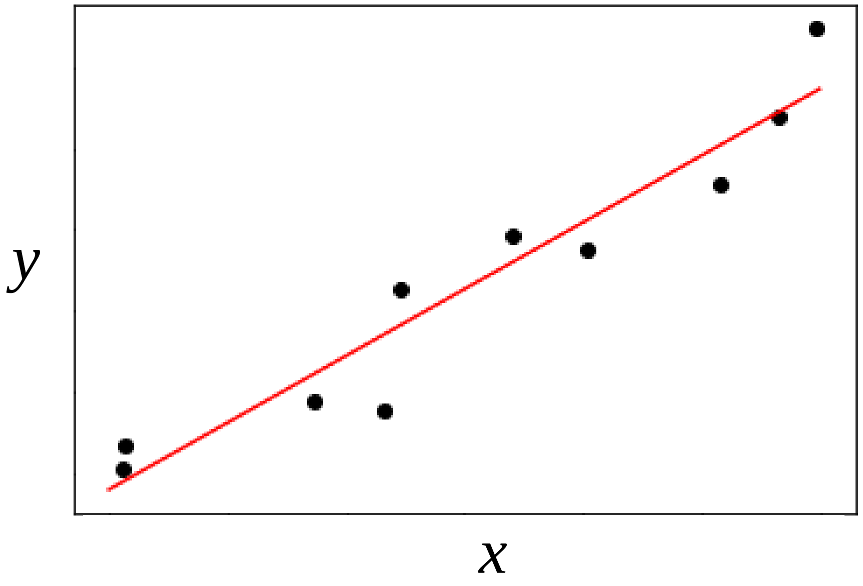
多项式回归
Polynomial Regression

逻辑回归
Logistic Regression

套索回归
Lasso Regression

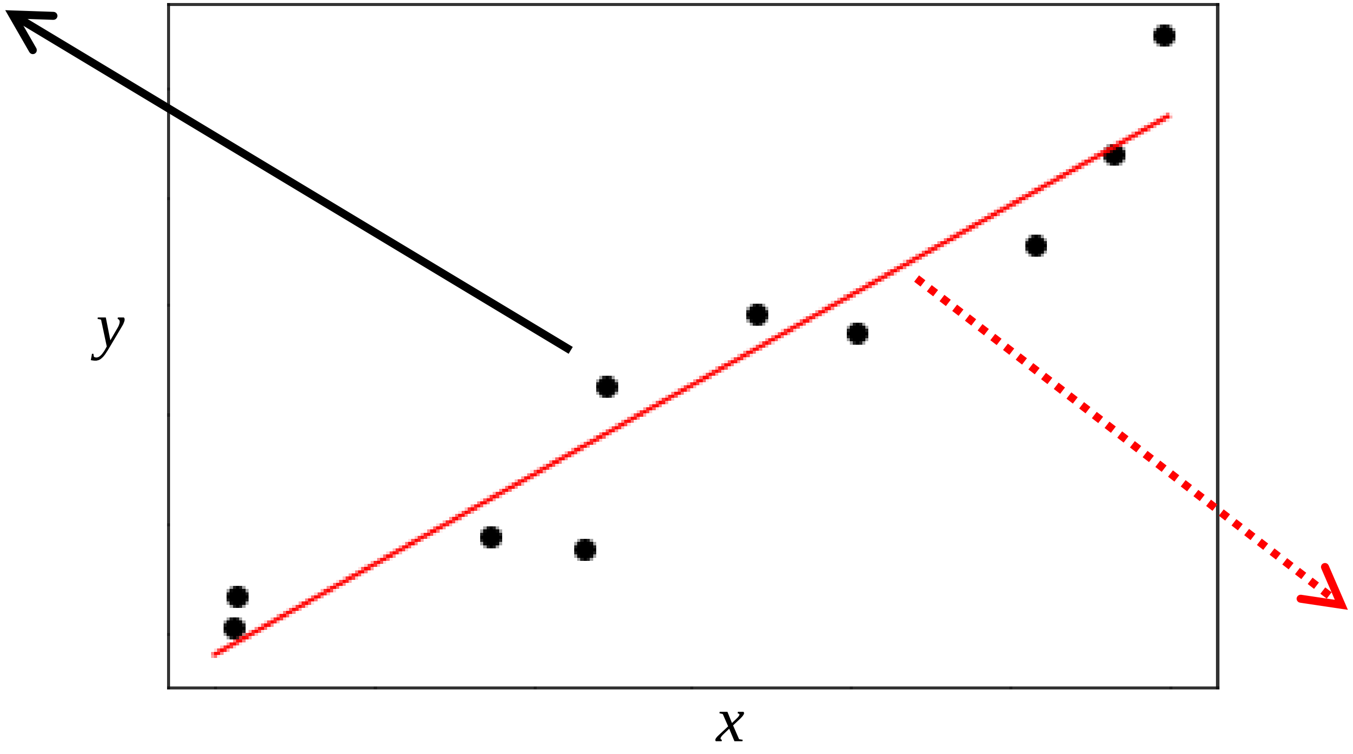
岭回归
Ridge Regression

…



回归算法原理

$$\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$$

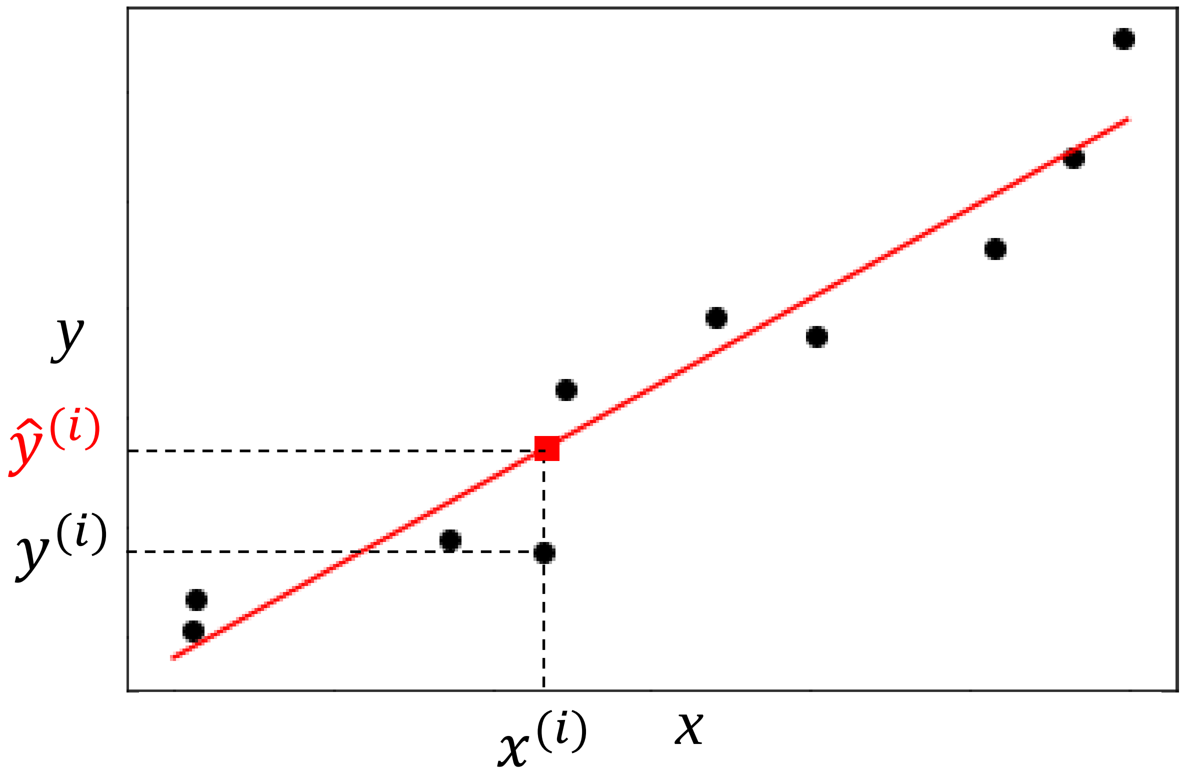


数据点（•）

映射关系（f）

$$\boxed{\hat{y} = f(x) = w_0 + w_1 x}$$

最小二乘法



$$\boxed{\hat{y}^{(i)} = f\left(x^{(i)}\right) = w_0 + w_1 x^{(i)}}$$

残差

$$\varepsilon_i = \hat{y}^{(i)} - y^{(i)}$$

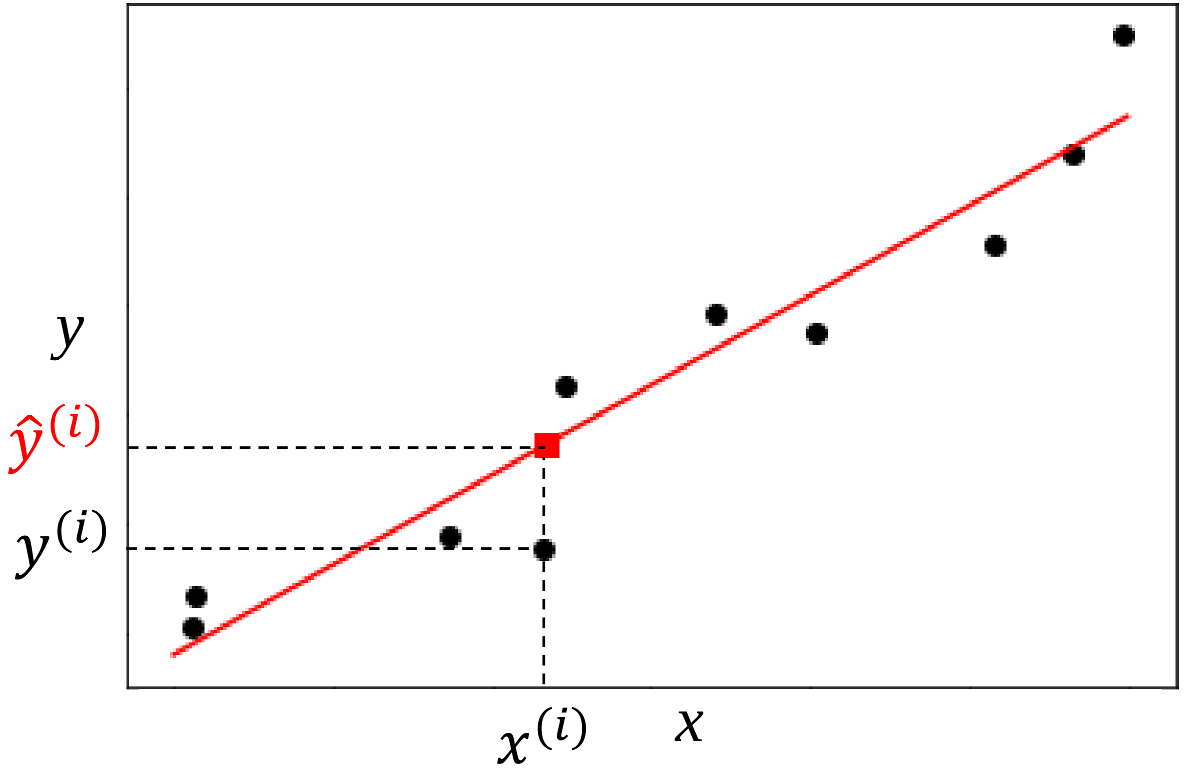
残差平方和

$$\sum {\varepsilon_i}^2 = \sum \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

最小二乘法原理

$$\min \sum \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

最小二乘法



$$\boxed{\hat{y}^{(i)} = f\left(x^{(i)}\right) = w_0 + w_1 x^{(i)}}$$

$$\min \sum \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$S = \sum \left(w_0 + w_1 x^{(i)} - y^{(i)}\right)^2$$

最小值处 S 对 w_0 与 w_1 的偏导数为0

$$\begin{cases} \dfrac{\partial S}{\partial w_0} = 2 \times \sum \left(w_0 + w_1 x^{(i)} - y^{(i)}\right) = 0 \\ \dfrac{\partial S}{\partial w_1} = 2 \times \sum \left(w_0 + w_1 x^{(i)} - y^{(i)}\right) x^{(i)} = 0 \end{cases}$$

最小二乘法

$$\begin{cases} \dfrac{\partial S}{\partial w_0} = 2 \times \sum\left(w_0 + w_1 x^{(i)} - y^{(i)}\right) = 0 \\ \dfrac{\partial S}{\partial w_1} = 2 \times \sum\left(w_0 + w_1 x^{(i)} - y^{(i)}\right) x^{(i)} = 0 \end{cases}$$

一元线性回归　　解简单的二元一次方程组

得到w_0与w_1

$$\boxed{f(x) = w_0 + w_1 x}$$

$$\boxed{w_0 = \bar{y} - w_1 \bar{x}}$$

$$\boxed{w_1 = \frac{\sum(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum(x^{(i)} - \bar{x})^2}}$$

$$\bar{x} = \frac{\sum x^{(i)}}{m} \qquad \bar{y} = \frac{\sum y^{(i)}}{m}$$

最小二乘法

n元情形

数据点

$$\left\{\left(x_1^{(1)},x_2^{(1)},\ldots,x_n^{(1)},y^{(1)}\right),\left(x_1^{(2)},x_2^{(2)},\ldots,x_n^{(2)},y^{(2)}\right),\cdots,\left(x_1^{(m)},x_2^{(m)},\ldots,x_n^{(m)},y^{(m)}\right)\right\}$$

映射关系

$$\boxed{f(x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n}$$

目标：寻找到一组$w(w_0,w_1,w_2,\cdots,w_n)$，
使得 $S(w) = \sum(\hat{y}^{(i)} - y^{(i)})^2$ 最小。

最小二乘法

n元情形

$$\hat{y}^{(1)} = w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)} + \cdots + w_n x_n^{(1)}$$

$$\hat{y}^{(2)} = w_0 + w_1 x_1^{(2)} + w_2 x_2^{(2)} + \cdots + w_n x_n^{(2)}$$

$$\cdots\cdots$$

$$\hat{y}^{(m)} = w_0 + w_1 x_1^{(m)} + w_2 x_2^{(m)} + \cdots + w_n x_n^{(m)}$$

令

$$\boldsymbol{X} = \begin{pmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}, \qquad \boldsymbol{w} = (w_1, w_2, \cdots, w_n)$$

$$\hat{\boldsymbol{y}} = \mathbf{X}\boldsymbol{w}^{\mathrm{T}} + \boldsymbol{w}_0 \qquad \hat{\boldsymbol{y}} = \mathbf{X}\boldsymbol{w}^{\mathrm{T}}$$

最小二乘法

n元情形

$$S(w) = \sum\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \sum\left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} - y^{(i)}\right)^2$$

$$= \|\mathbf{X}\boldsymbol{w}^T - \boldsymbol{y}\|^2$$

同样的，求导

$$\frac{\partial S(\boldsymbol{w})}{\partial \boldsymbol{w}^T} = 2\sum\sum x_j^{(i)}(w_j x_j^{(i)} - y^{(i)})$$

$$= 2\mathbf{X}^T(\mathbf{X}\boldsymbol{w}^T - \boldsymbol{y})$$

令偏导数为0

$$\boldsymbol{w}^{\boldsymbol{T}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

正规方程

The Normal Equation

目录

1. 算法简介

2. 回归算法

3. 双金属吸附中的回归

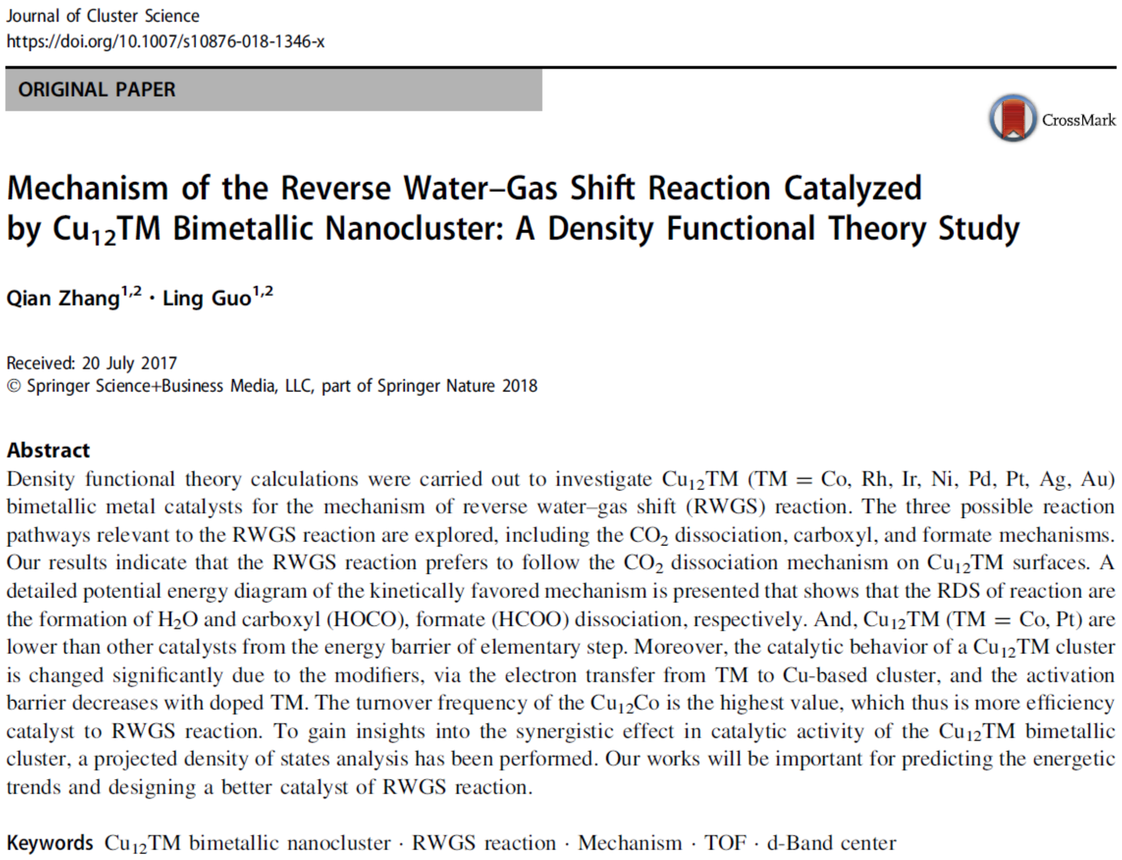
4. 非线性回归求速率常数

5. 模型评价

6. 决策树分类算法

7. 支持向量机

文献中的线性回归



Journal of Cluster Science
https://doi.org/10.1007/s10876-018-1346-x

ORIGINAL PAPER

CrossMark

Mechanism of the Reverse Water–Gas Shift Reaction Catalyzed by Cu_{12}TM Bimetallic Nanocluster: A Density Functional Theory Study

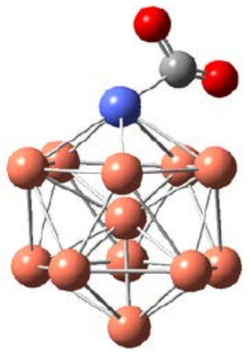
Qian Zhang[1,2] · Ling Guo[1,2]

Received: 20 July 2017

Abstract

Density functional theory calculations were carried out to investigate Cu_{12}TM (TM = Co, Rh, Ir, Ni, Pd, Pt, Ag, Au) bimetallic metal catalysts for the mechanism of reverse water–gas shift (RWGS) reaction. The three possible reaction pathways relevant to the RWGS reaction are explored, including the CO_2 dissociation, carboxyl, and formate mechanisms. Our results indicate that the RWGS reaction prefers to follow the CO_2 dissociation mechanism on Cu_{12}TM surfaces. A detailed potential energy diagram of the kinetically favored mechanism is presented that shows that the RDS of reaction are the formation of H_2O and carboxyl (HOCO), formate (HCOO) dissociation, respectively. And, Cu_{12}TM (TM = Co, Pt) are lower than other catalysts from the energy barrier of elementary step. Moreover, the catalytic behavior of a Cu_{12}TM cluster is changed significantly due to the modifiers, via the electron transfer from TM to Cu-based cluster, and the activation barrier decreases with doped TM. The turnover frequency of the Cu_{12}Co is the highest value, which thus is more efficiency catalyst to RWGS reaction. To gain insights into the synergistic effect in catalytic activity of the Cu_{12}TM bimetallic cluster, a projected density of states analysis has been performed. Our works will be important for predicting the energetic trends and designing a better catalyst of RWGS reaction.

Keywords Cu_{12}TM bimetallic nanocluster · RWGS reaction · Mechanism · TOF · d-Band center

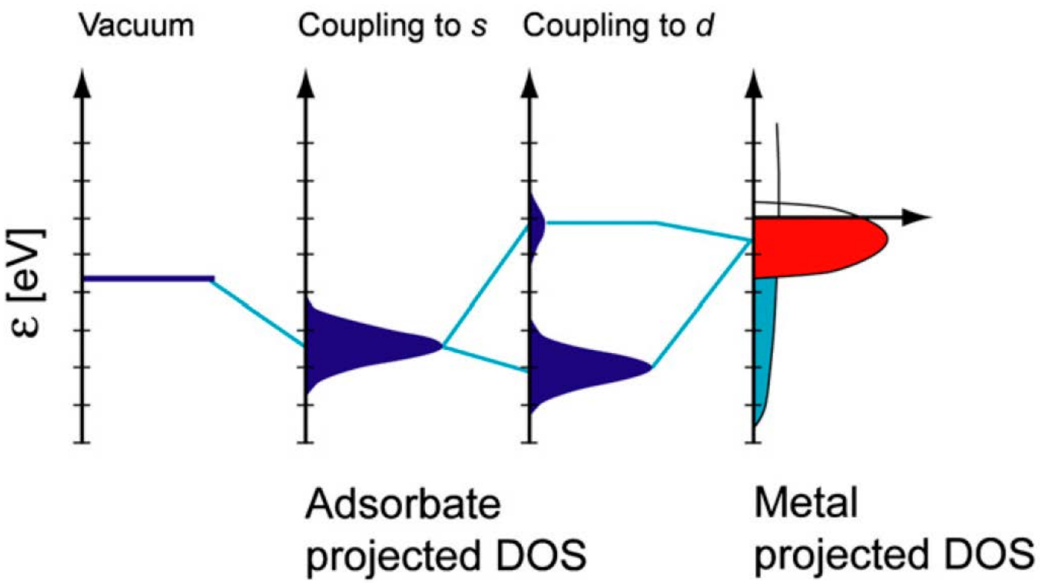


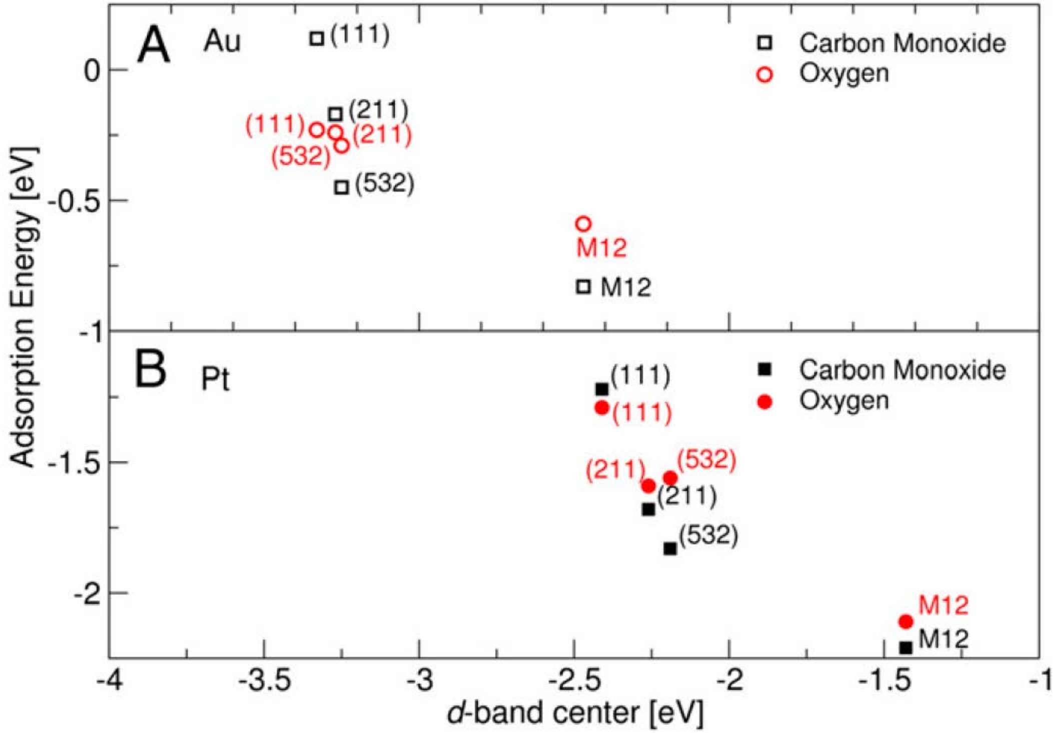
Cu_{12}TM-CO2*

Q. Zhang and L. Guo, J. Clust. Sci. 2018, 29, 867.

d带中心理论

CO、O_2在Au与Pt上的吸附能





J. K. Nørskov, T. Bligaardc, *et. al. Proc. Natl. Acad. Sci* **2011**, 108, 937.

d带中心与吸附能

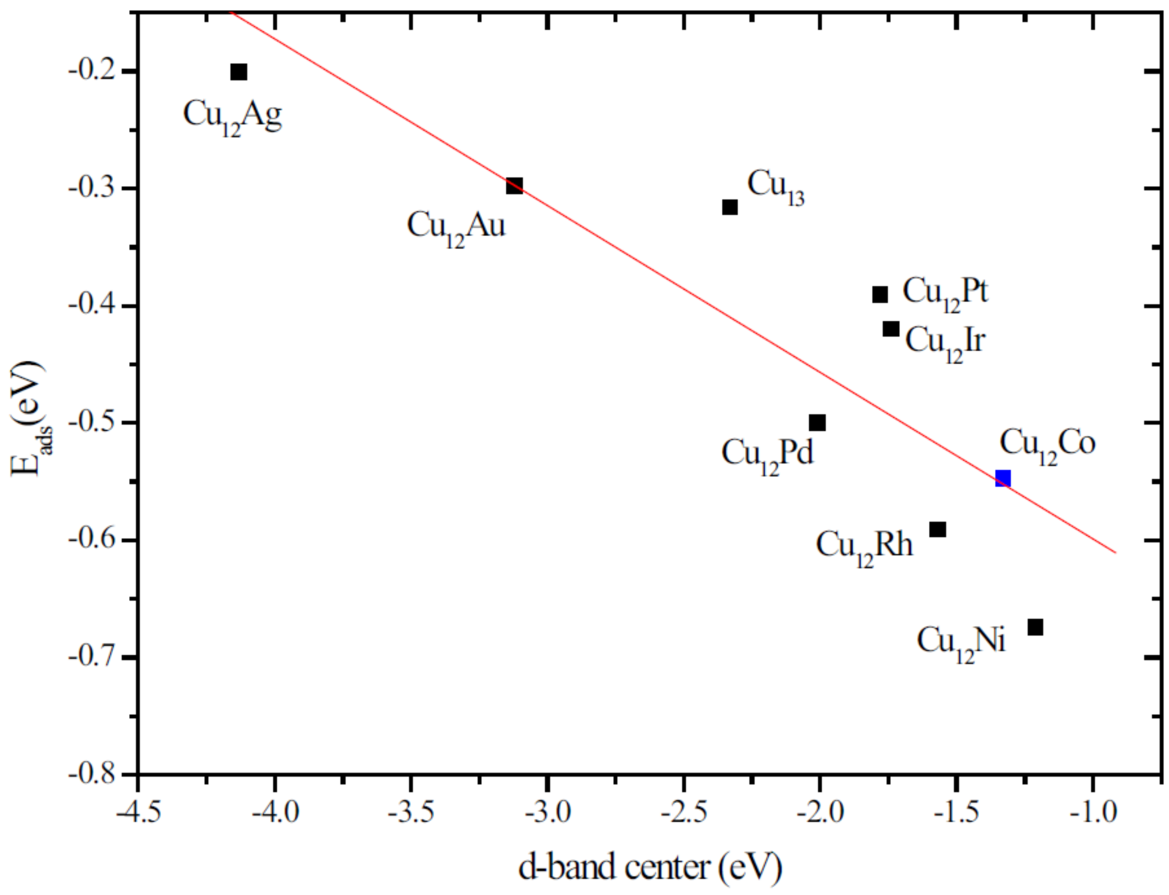
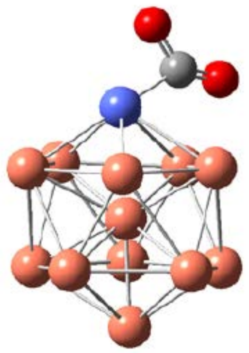
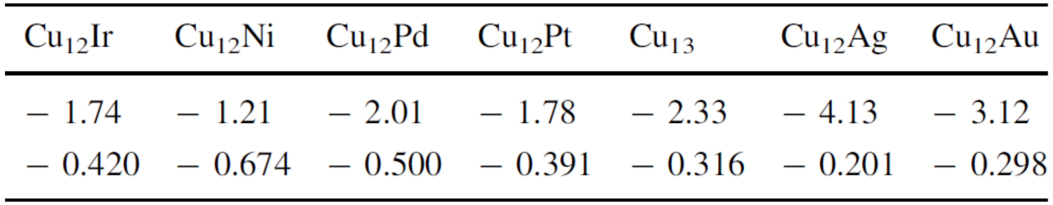


Fig. 6 Relationship between d-band center of TM metal atom and adsorption energy of CO_2 molecular



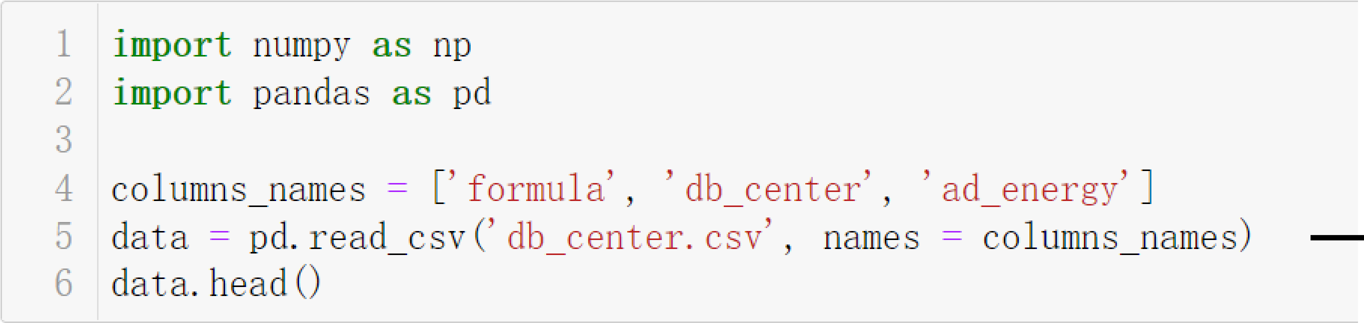
$Cu_{12}TM$-CO2*

d带中心 ⟷ CO_2吸附能



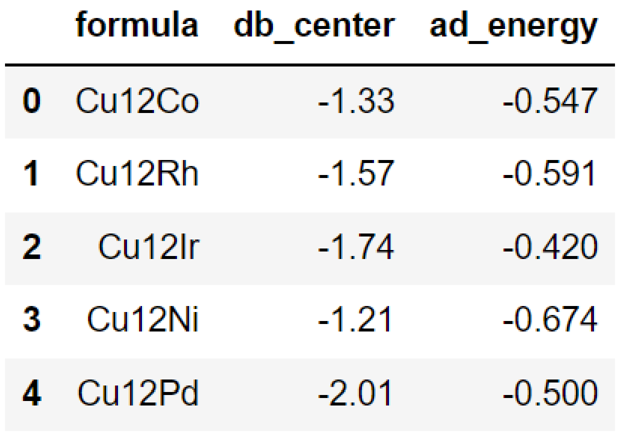
$Cu_{12}Ir$	$Cu_{12}Ni$	$Cu_{12}Pd$	$Cu_{12}Pt$	Cu_{13}	$Cu_{12}Ag$	$Cu_{12}Au$
− 1.74	− 1.21	− 2.01	− 1.78	− 2.33	− 4.13	− 3.12
− 0.420	− 0.674	− 0.500	− 0.391	− 0.316	− 0.201	− 0.298

双金属吸附中的回归

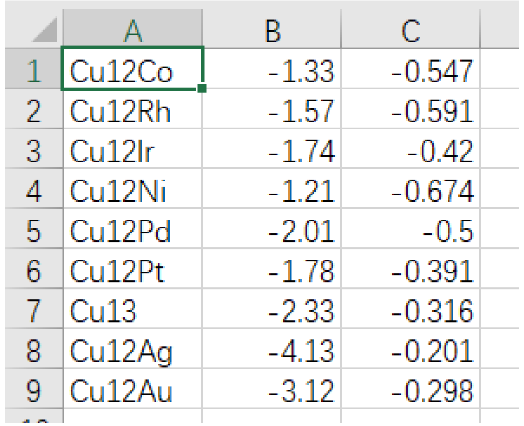


```python
import numpy as np
import pandas as pd

columns_names = ['formula', 'db_center', 'ad_energy']
data = pd.read_csv('db_center.csv', names = columns_names)
data.head()
```

→ 读取文件

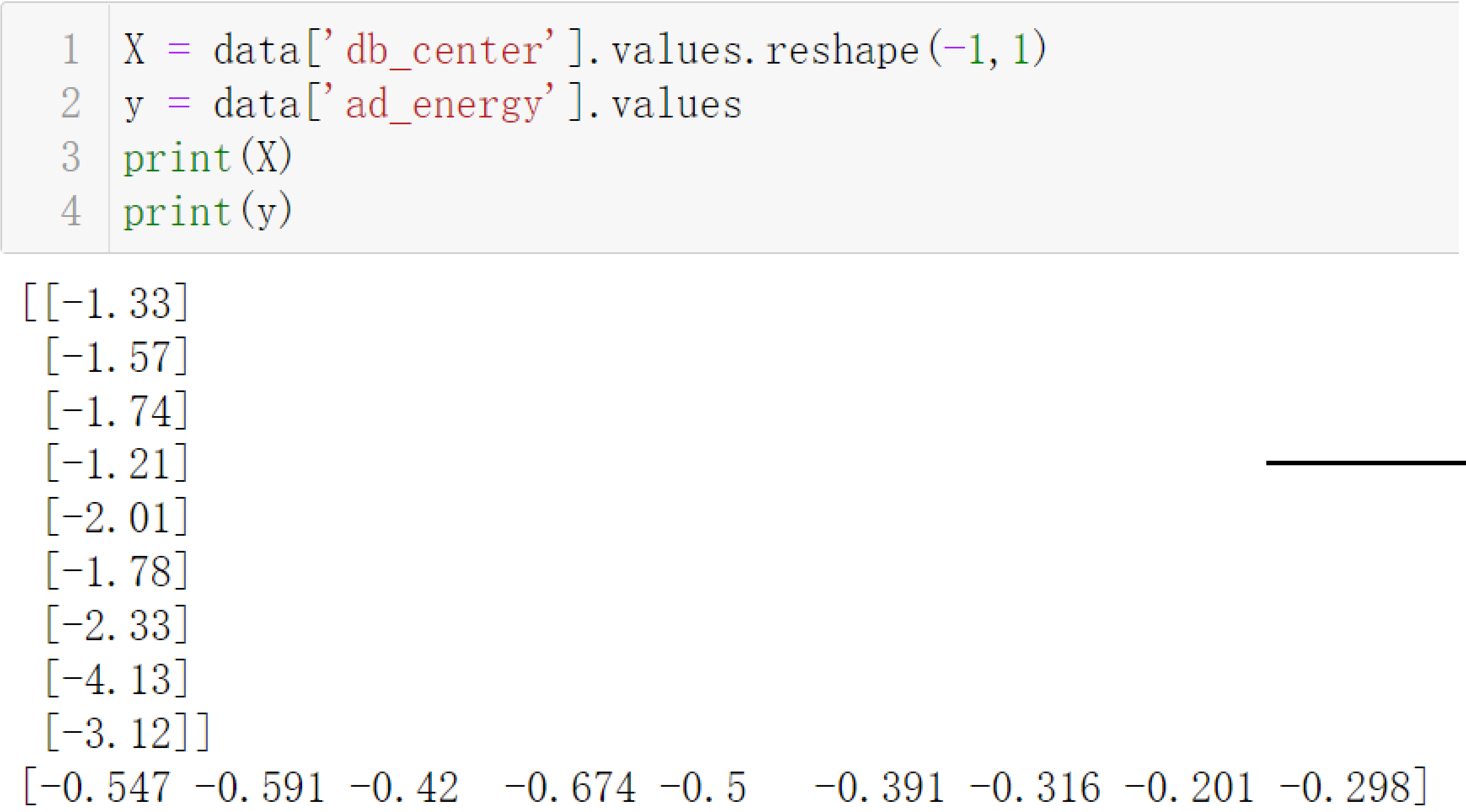


	formula	db_center	ad_energy
0	Cu12Co	-1.33	-0.547
1	Cu12Rh	-1.57	-0.591
2	Cu12Ir	-1.74	-0.420
3	Cu12Ni	-1.21	-0.674
4	Cu12Pd	-2.01	-0.500



	A	B	C
1	Cu12Co	-1.33	-0.547
2	Cu12Rh	-1.57	-0.591
3	Cu12Ir	-1.74	-0.42
4	Cu12Ni	-1.21	-0.674
5	Cu12Pd	-2.01	-0.5
6	Cu12Pt	-1.78	-0.391
7	Cu13	-2.33	-0.316
8	Cu12Ag	-4.13	-0.201
9	Cu12Au	-3.12	-0.298

双金属吸附中的回归

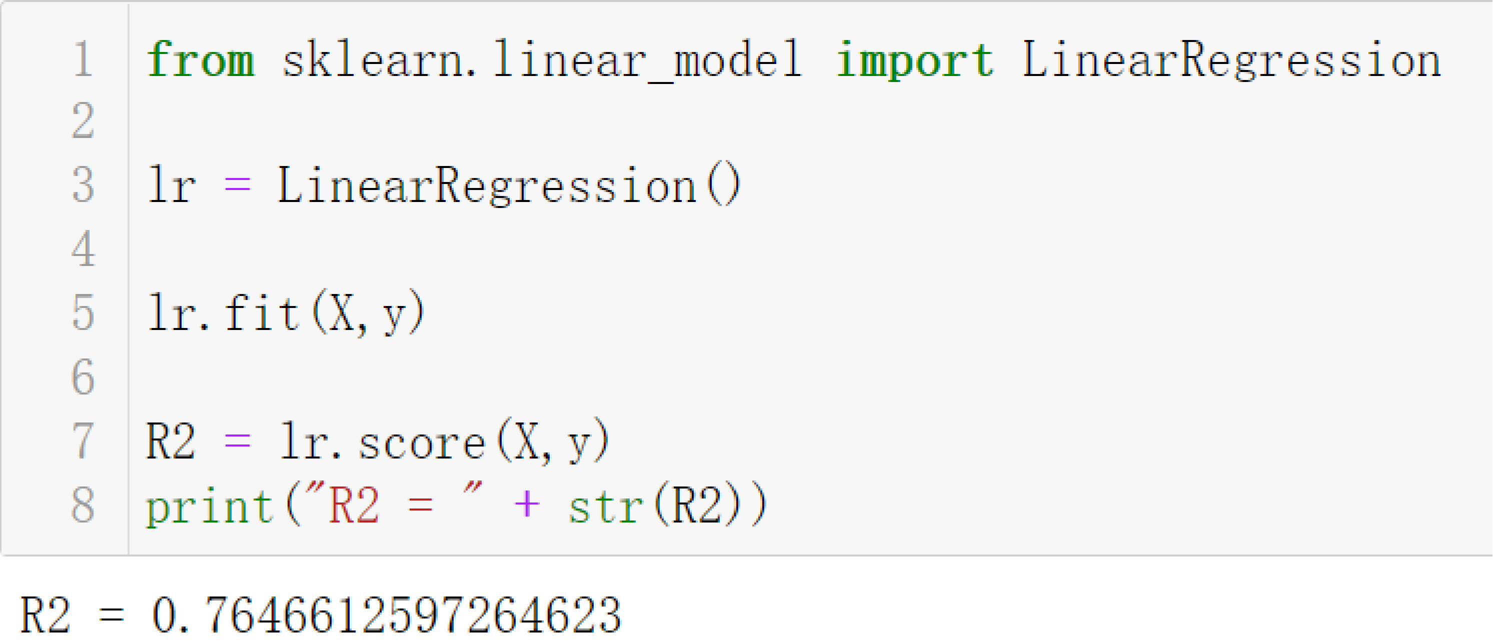


```python
X = data['db_center'].values.reshape(-1,1)
y = data['ad_energy'].values
print(X)
print(y)
```

```
[[-1.33]
 [-1.57]
 [-1.74]
 [-1.21]
 [-2.01]
 [-1.78]
 [-2.33]
 [-4.13]
 [-3.12]]
[-0.547 -0.591 -0.42  -0.674 -0.5   -0.391 -0.316 -0.201 -0.298]
```

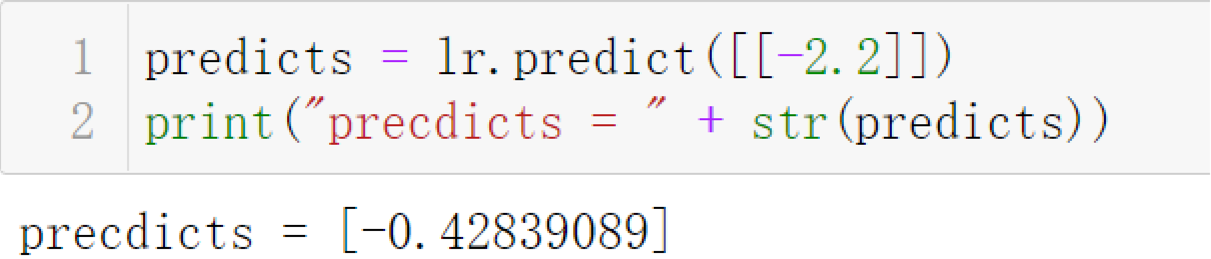
→ 查看文件

双金属吸附中的回归



```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(X, y)

R2 = lr.score(X, y)
print("R2 = " + str(R2))
```

```
R2 = 0.7646612597264623
```



```
predicts = lr.predict([[-2.2]])
print("precdicts = " + str(predicts))
```

```
precdicts = [-0.42839089]
```

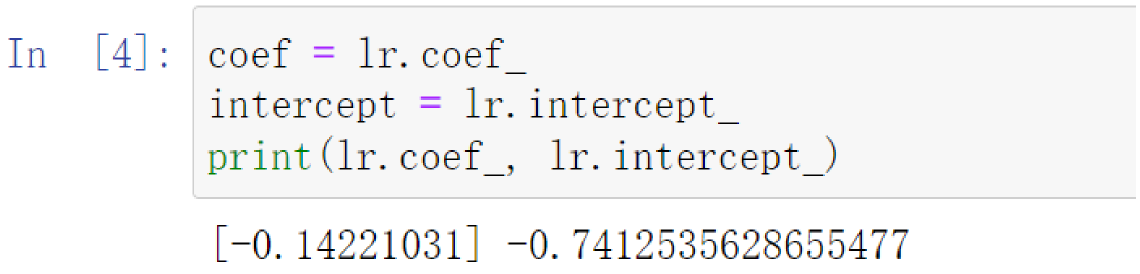
→ 预测

双金属吸附中的回归

输出斜率与截距的方法

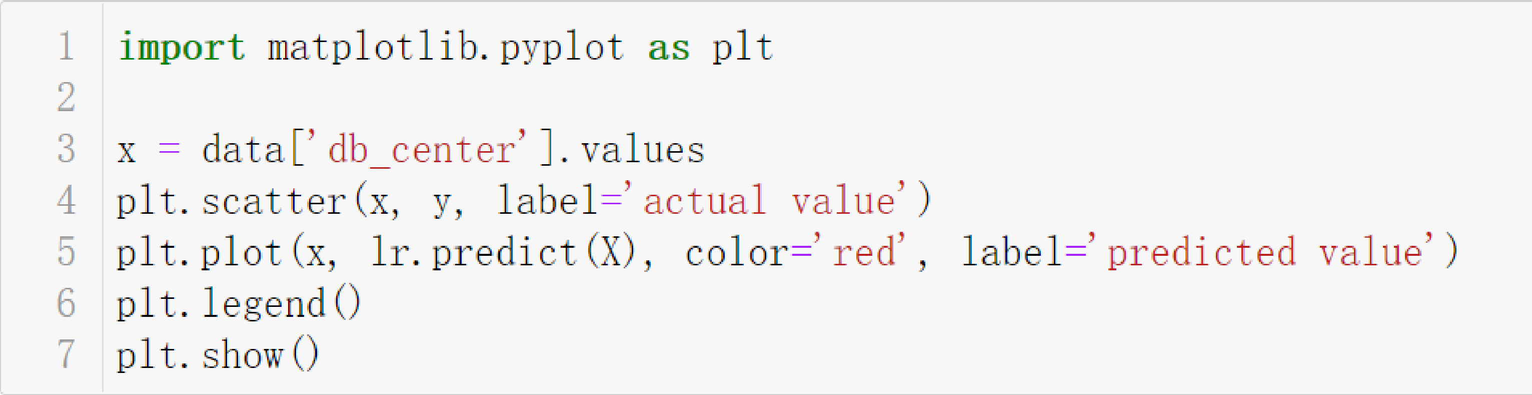
斜率　coef = lr.coef_

截距　intercept = lr.intercept_

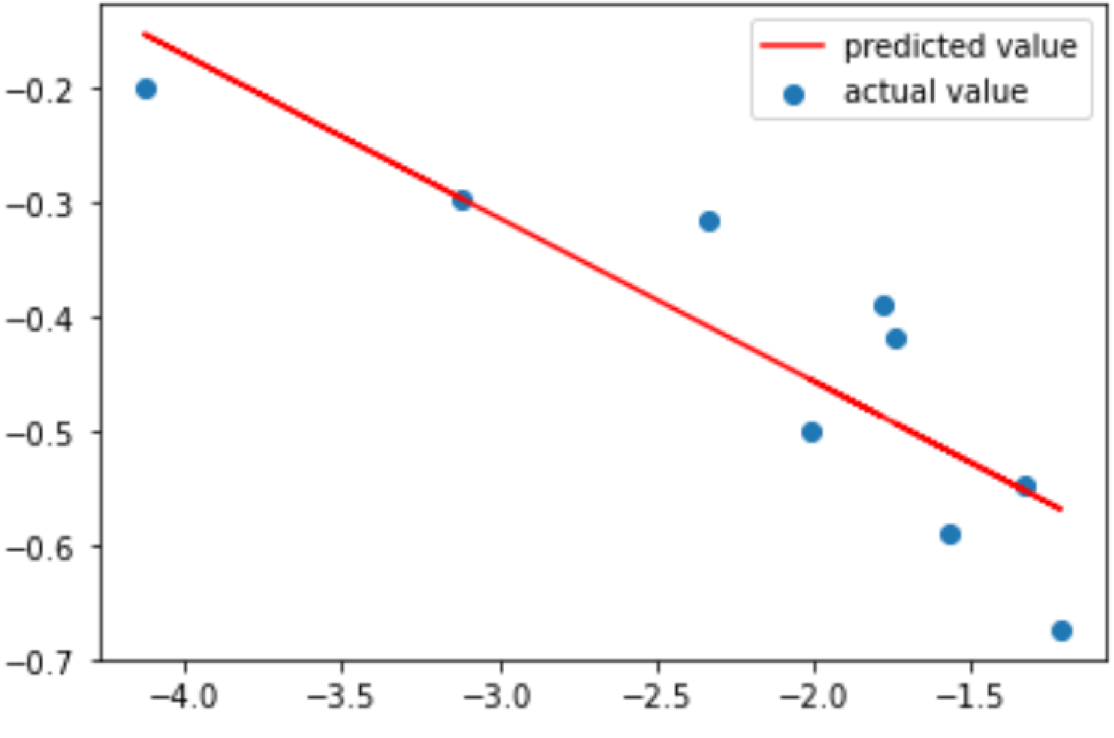


```
In  [4]: coef = lr.coef_
         intercept = lr.intercept_
         print(lr.coef_, lr.intercept_)

         [-0.14221031] -0.7412535628655477
```

双金属吸附中的回归



```
import matplotlib.pyplot as plt

x = data[’db_center’].values
plt.scatter(x, y, label=’actual value’)
plt.plot(x, lr.predict(X), color=’red’, label=’predicted value’)
plt.legend()
plt.show()
```



目录

1. 算法简介
2. 回归算法
3. 双金属吸附中的回归
4. 非线性回归求速率常数
5. 模型评价
6. 决策树分类算法
7. 支持向量机

非线性回归背景：连续反应动力学

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C}$$

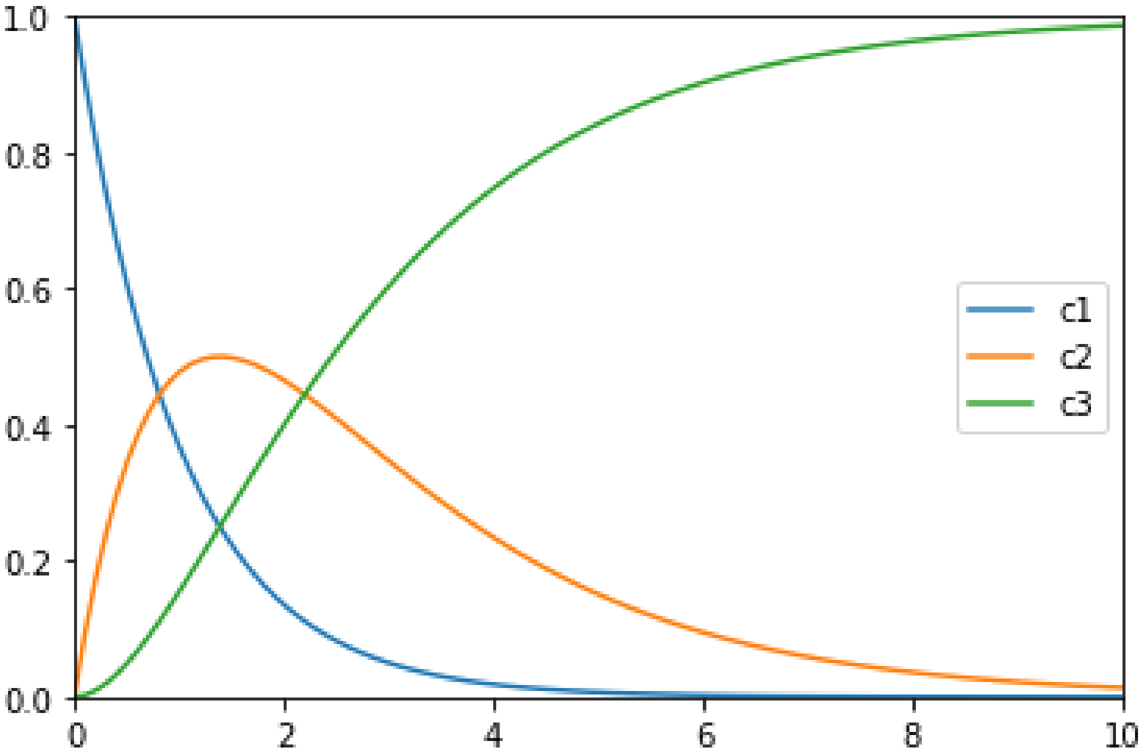
$$\frac{dc_A}{dt} = -k_1 c_A \qquad\qquad c_A = c_{A0} e^{k_1 t}$$

$$\frac{dc_B}{dt} = k_1 c_A - k_2 c_B \quad \Rightarrow \quad c_B = \frac{k_1 c_{A0}}{k_2 - k_1}\left(e^{-k_1 t} - e^{-k_2 t}\right)$$

$$\frac{dc_c}{dt} = k_2 c_B \qquad\qquad c_C = c_{A0}\left(1 - \frac{k_2}{k_2 - k_1} e^{-k_1 t} - \frac{k_1}{k_2 - k_1} e^{-k_2 t}\right)$$

连续反应动力学

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$



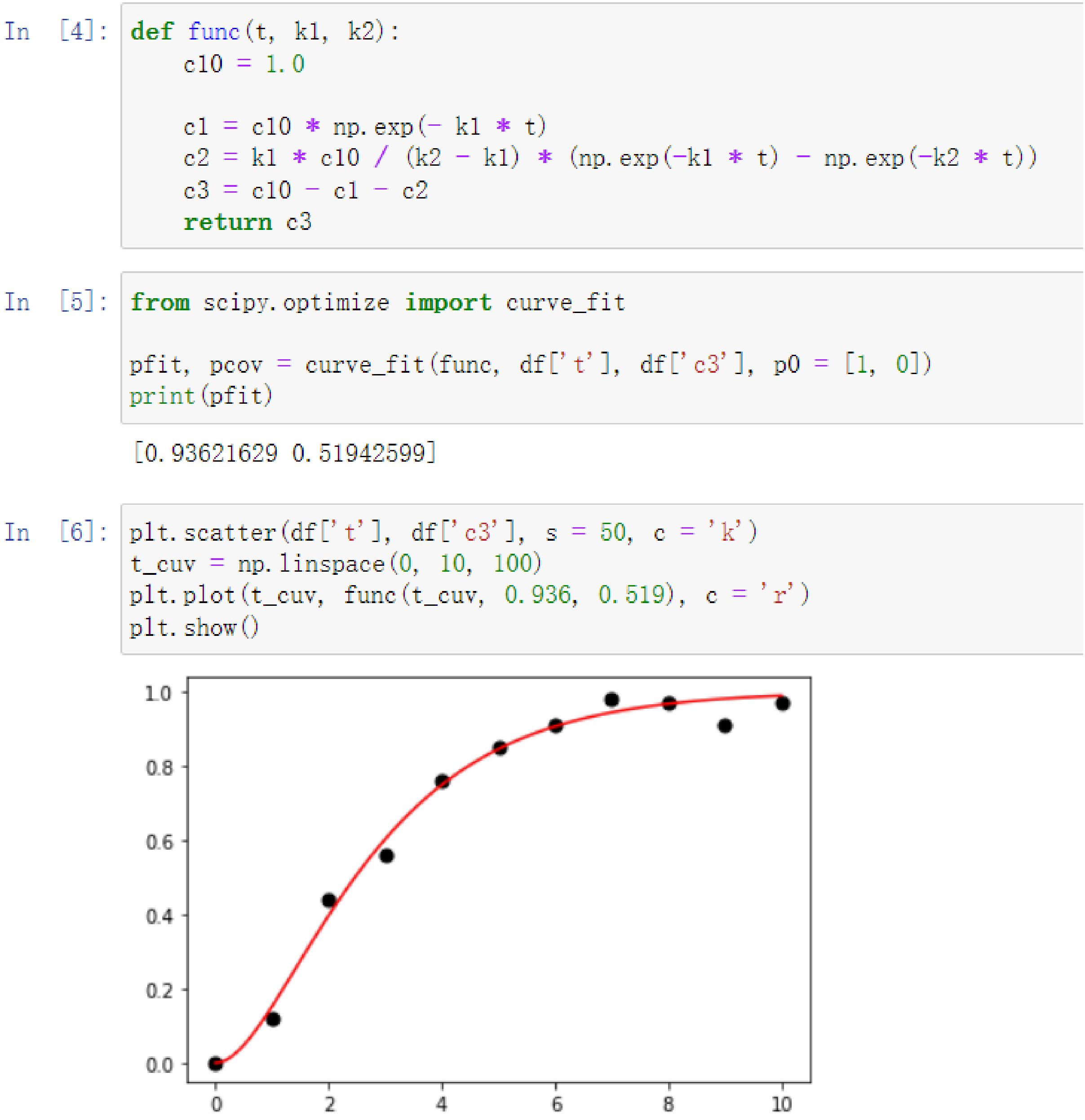
$$c_{A0} = 1.0\, mol/L$$

$$k_1 = 1.0\, mol/L/s$$

$$k_2 = 0.5\, mol/L/s$$

$$t = 10\ s$$

实操：反应速率的确定



```
In [4]: def func(t, k1, k2):
            c10 = 1.0

            c1 = c10 * np.exp(- k1 * t)
            c2 = k1 * c10 / (k2 - k1) * (np.exp(-k1 * t) - np.exp(-k2 * t))
            c3 = c10 - c1 - c2
            return c3
```

```
In [5]: from scipy.optimize import curve_fit

        pfit, pcov = curve_fit(func, df['t'], df['c3'], p0 = [1, 0])
        print(pfit)

        [0.93621629 0.51942599]
```

```
In [6]: plt.scatter(df['t'], df['c3'], s = 50, c = 'k')
        t_cuv = np.linspace(0, 10, 100)
        plt.plot(t_cuv, func(t_cuv, 0.936, 0.519), c = 'r')
        plt.show()
```

非线性回归

Python提供了自定义函数回归的方式

curve_fit()

```
def func(x, p1, p2):
    ...

pfit, pcov = curve_fit(func, x_val, y_val)
```

np数组，协方差矩阵

np数组，拟合参数

$$\sigma(x,y)=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$$

目录

1. 算法简介

2. 回归算法

3. 双金属吸附中的回归

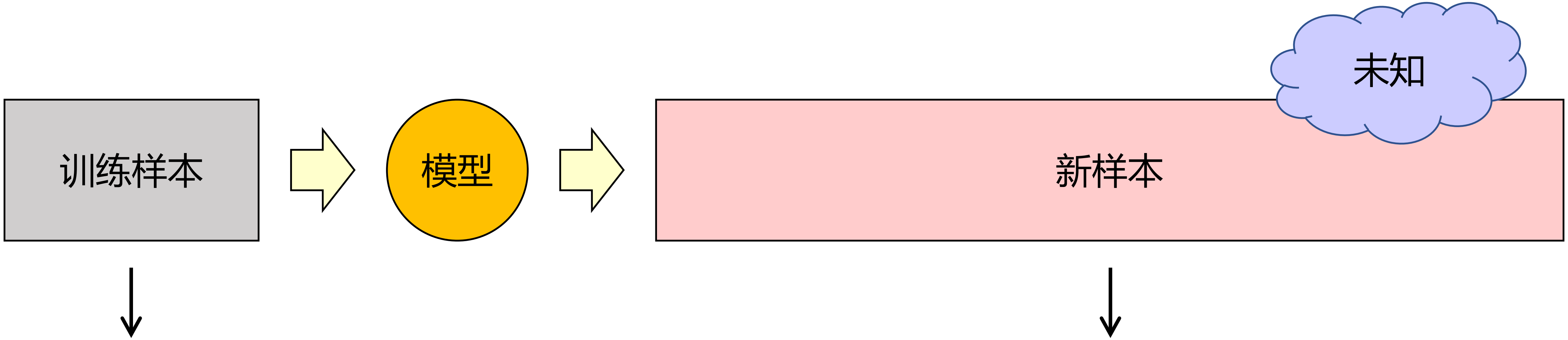
4. 非线性回归求速率常数

5. 模型评价

6. 决策树分类算法

7. 支持向量机

误差

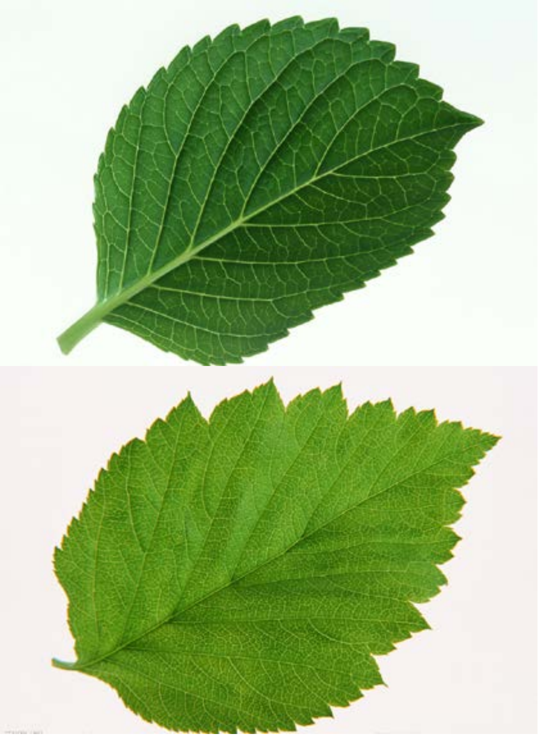


训练误差（training error）

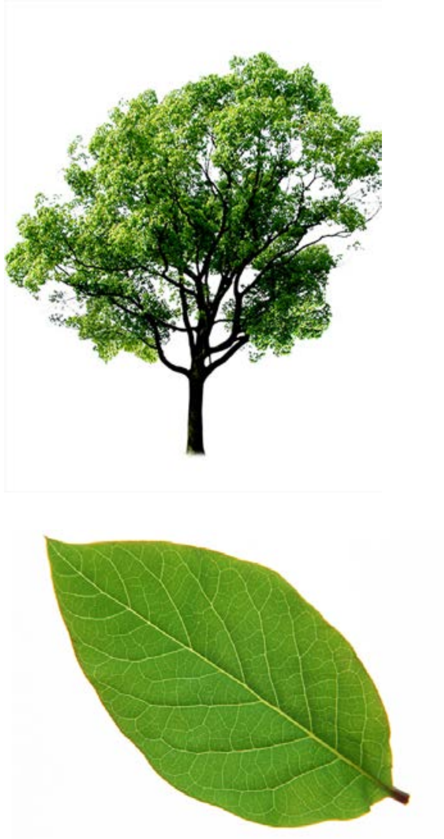
泛化误差（generalization error）

欠拟合与过拟合

判断图片中是否是树叶



树叶训练集

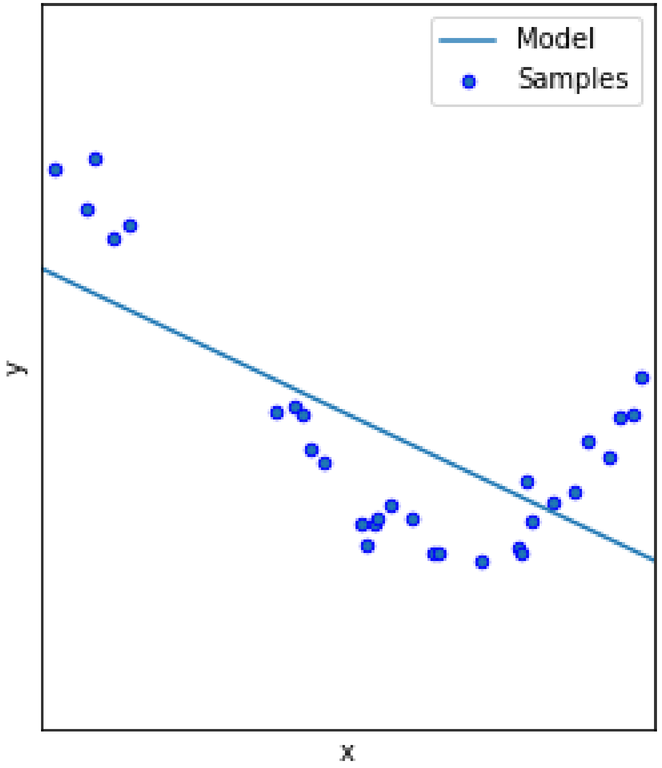


测试集

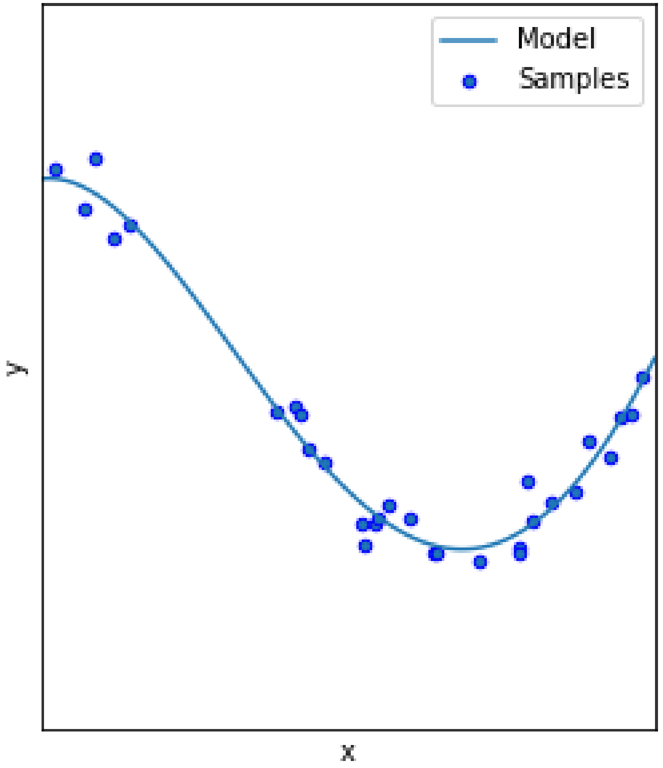
欠拟合(Underfitting)
结果：是树叶
绿色的就是树叶

过拟合(Overfitting)
结果：不是树叶
有锯齿的才是树叶

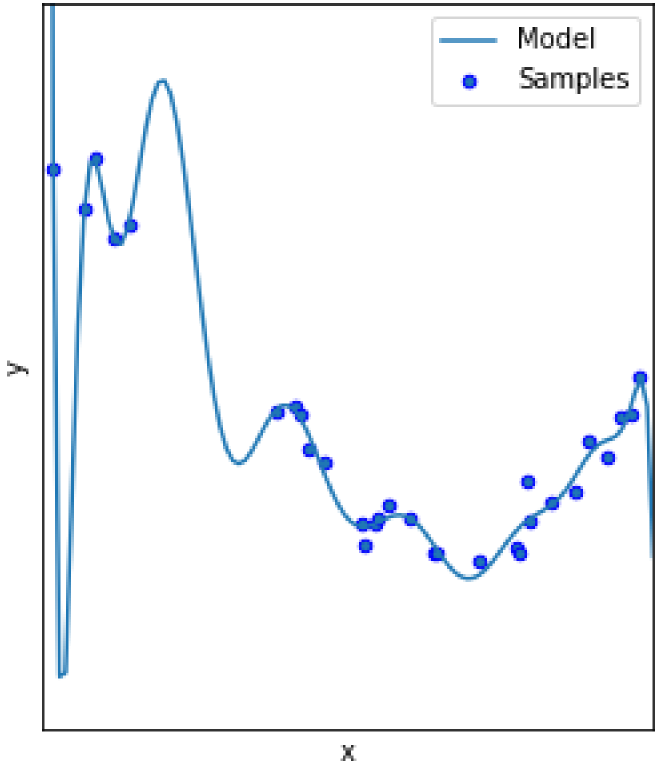
欠拟合与过拟合



欠拟合

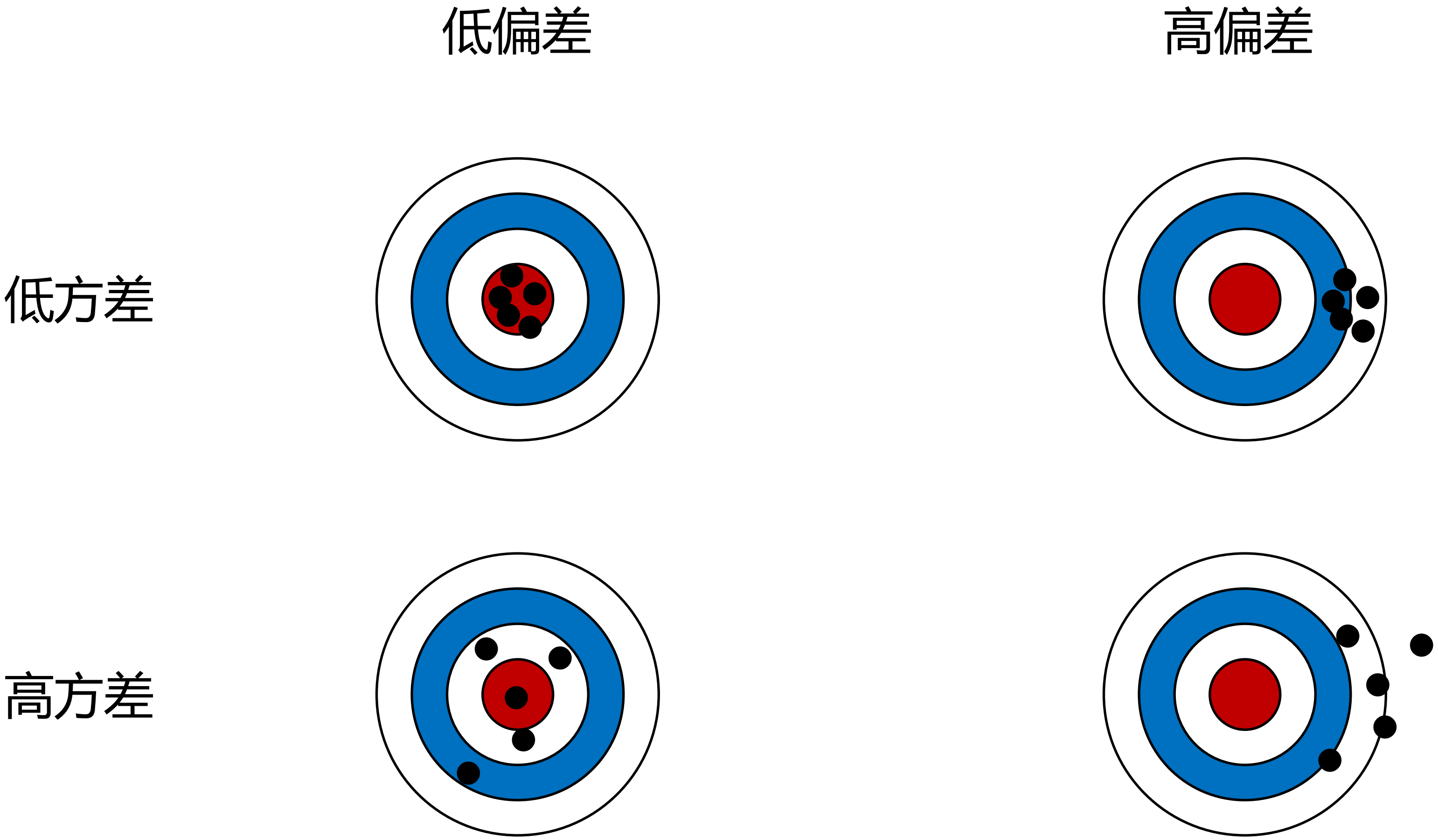


准确的模型



过拟合

偏差与方差



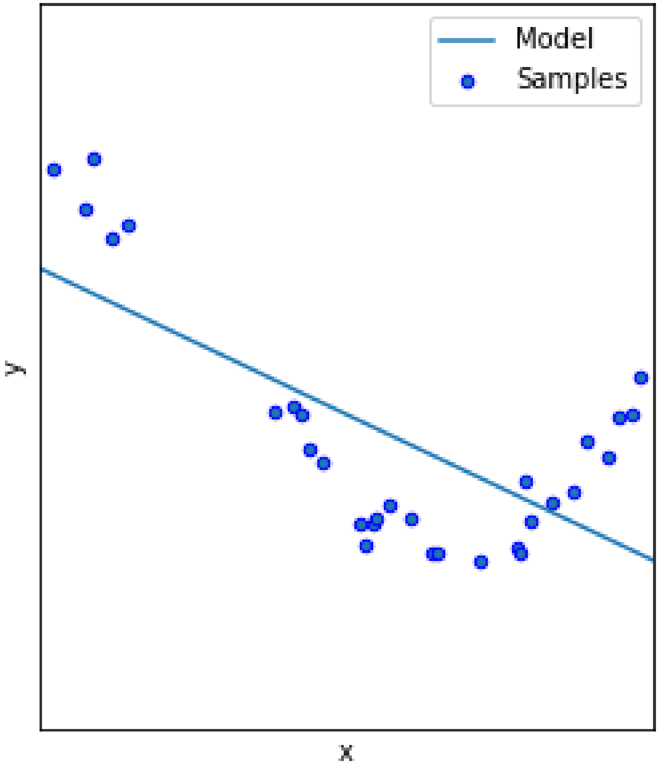
低偏差

高偏差

低方差

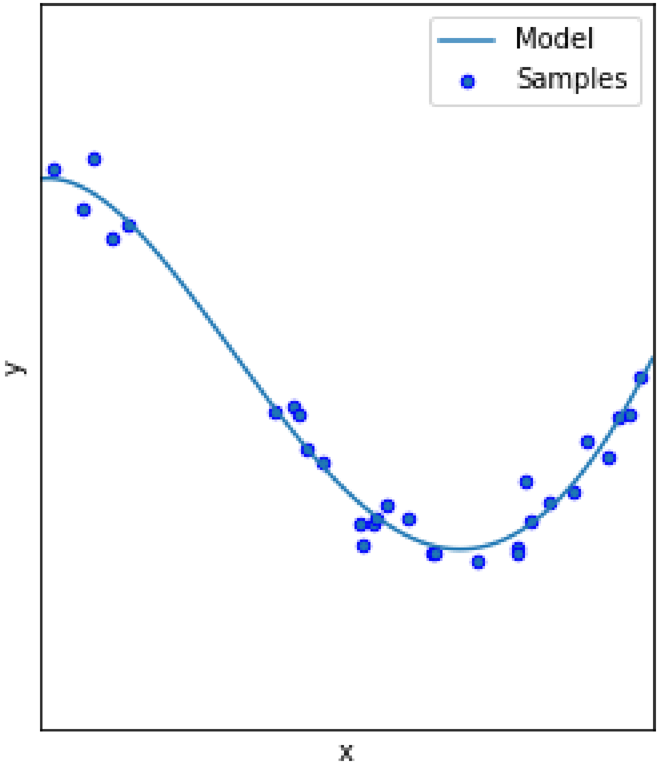
高方差

准确的概念

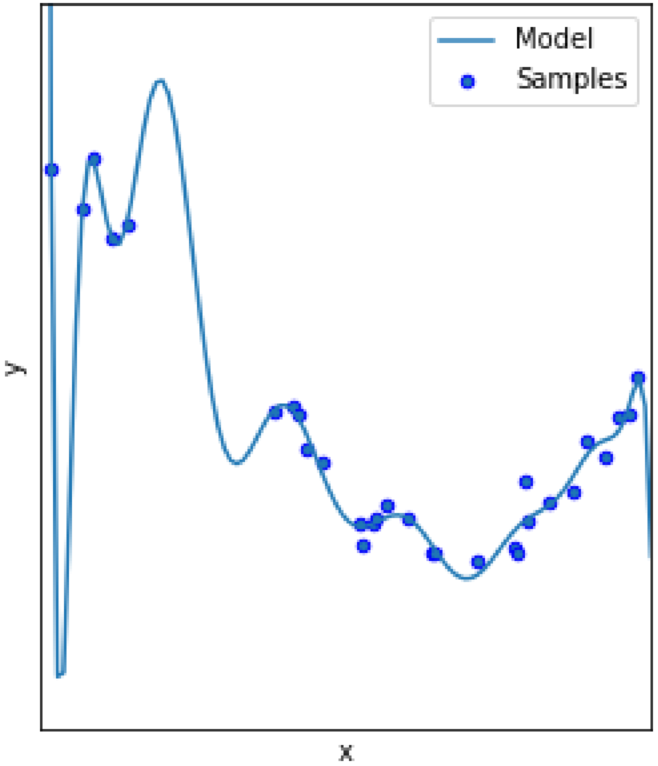


欠拟合

高偏差，低方差
High bias, Low variance



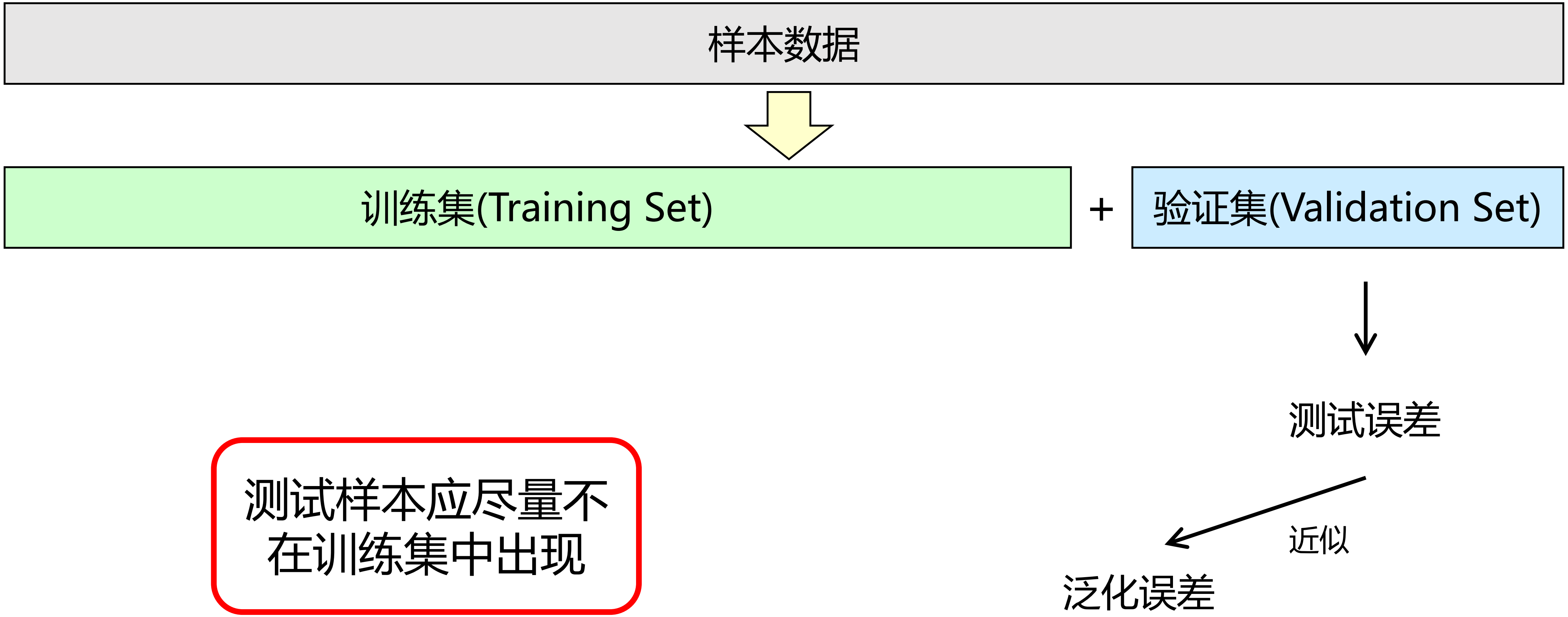
权衡偏差方差
Bias Variance Trade-off



过拟合

低偏差，高方差
Low bias, High variance

训练集与测试集



样本数据

训练集(Training Set) + 验证集(Validation Set)

测试误差

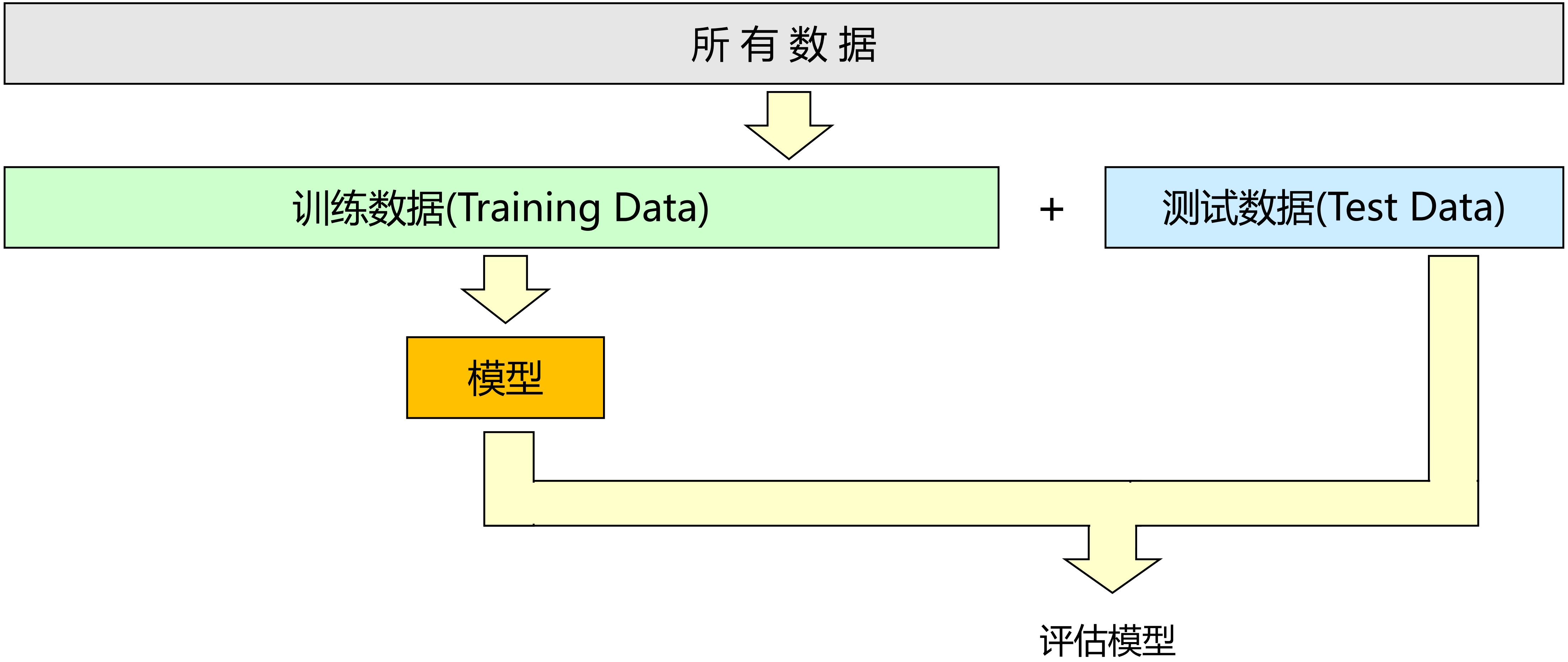
近似

泛化误差

测试样本应尽量不在训练集中出现

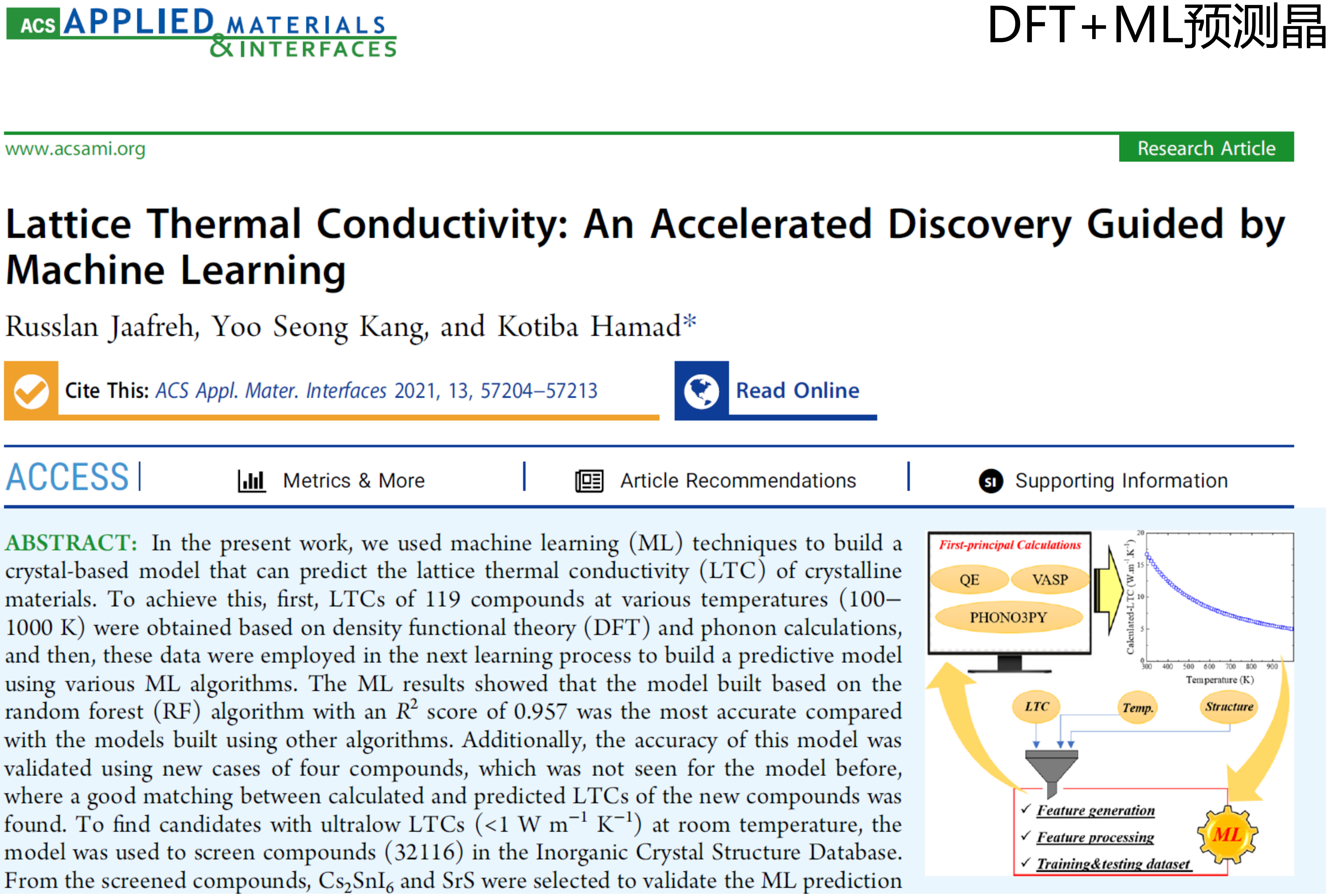
留出法

留出法（hold-out）



交叉验证示例

DFT+ML预测晶格热导率



ACS APPLIED MATERIALS & INTERFACES

www.acsami.org

Research Article

Lattice Thermal Conductivity: An Accelerated Discovery Guided by Machine Learning

Russlan Jaafreh, Yoo Seong Kang, and Kotiba Hamad*

Cite This: *ACS Appl. Mater. Interfaces* 2021, 13, 57204−57213

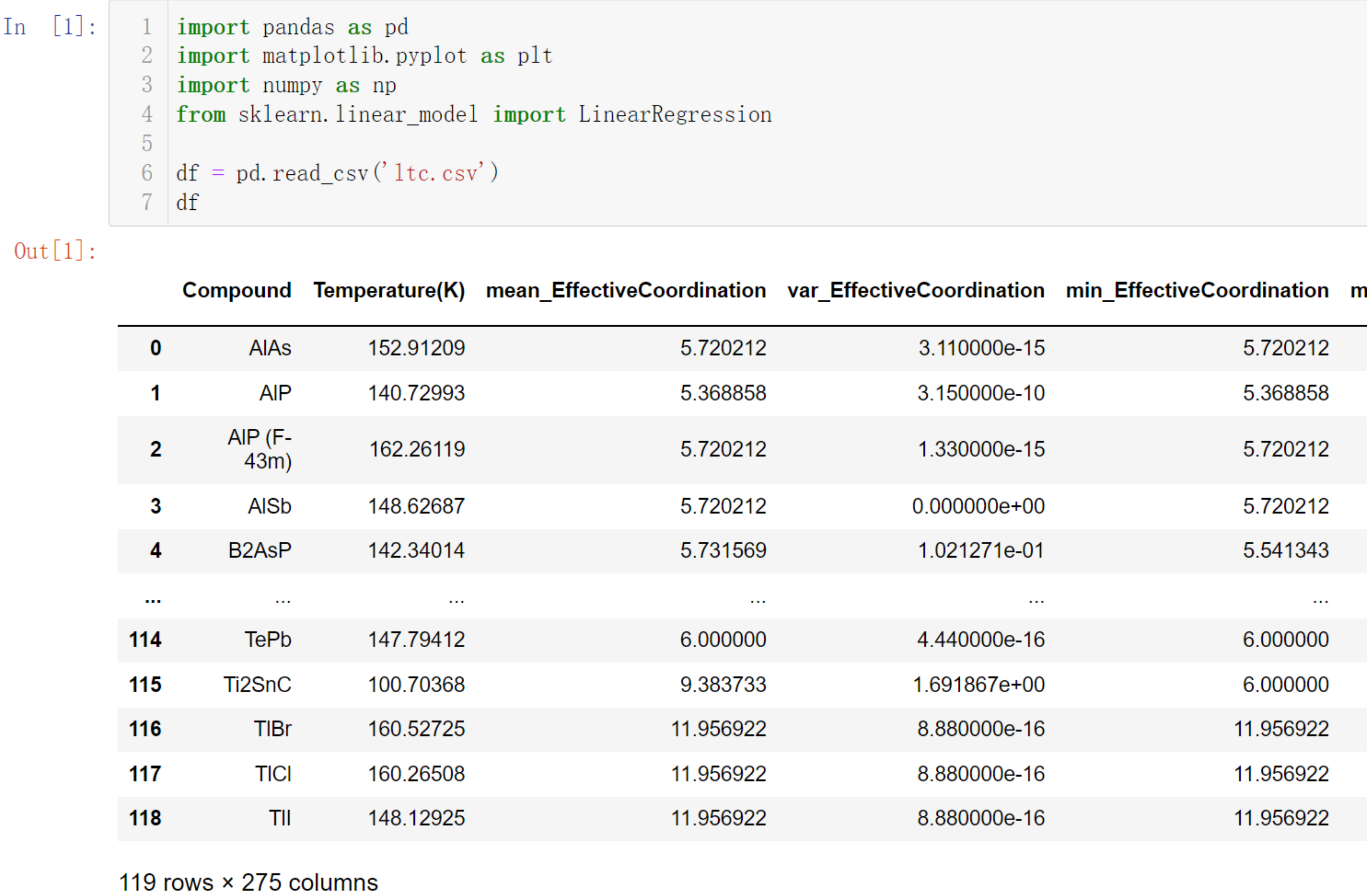
Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: In the present work, we used machine learning (ML) techniques to build a crystal-based model that can predict the lattice thermal conductivity (LTC) of crystalline materials. To achieve this, first, LTCs of 119 compounds at various temperatures (100−1000 K) were obtained based on density functional theory (DFT) and phonon calculations, and then, these data were employed in the next learning process to build a predictive model using various ML algorithms. The ML results showed that the model built based on the random forest (RF) algorithm with an R^2 score of 0.957 was the most accurate compared with the models built using other algorithms. Additionally, the accuracy of this model was validated using new cases of four compounds, which was not seen for the model before, where a good matching between calculated and predicted LTCs of the new compounds was found. To find candidates with ultralow LTCs (<1 W m^{-1} K^{-1}) at room temperature, the model was used to screen compounds (32116) in the Inorganic Crystal Structure Database. From the screened compounds, Cs_2SnI_6 and SrS were selected to validate the ML prediction

示例

整个模型的拟合结果



In [1]:

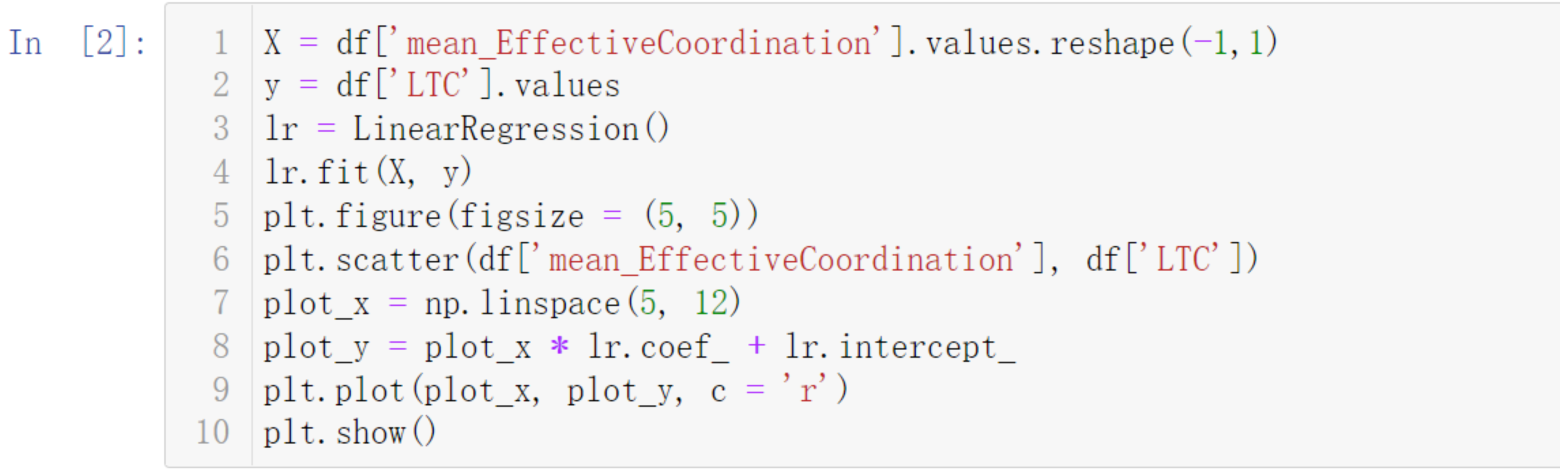
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

df = pd.read_csv('ltc.csv')
df
```

Out[1]:

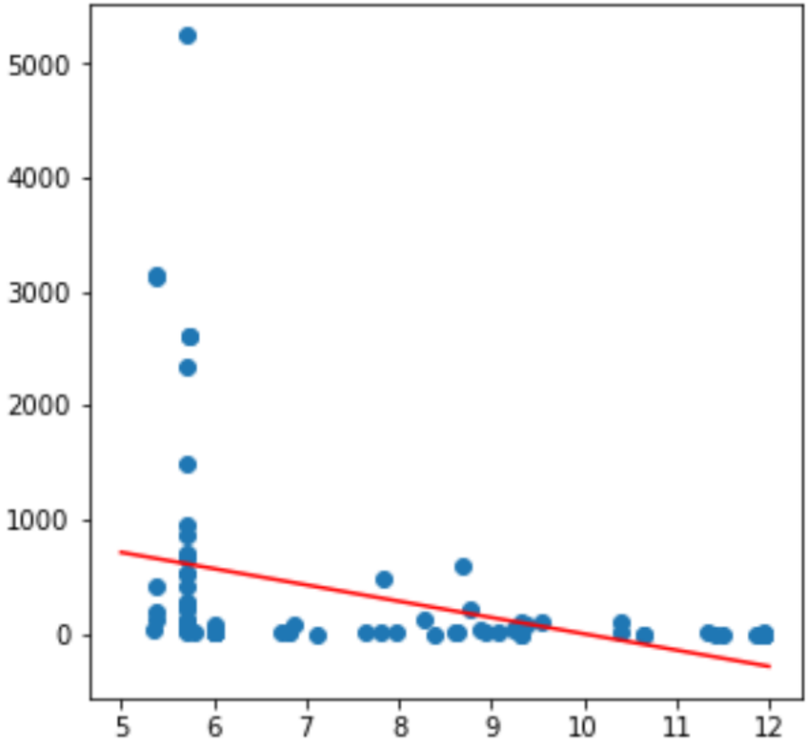
	Compound	Temperature(K)	mean_EffectiveCoordination	var_EffectiveCoordination	min_EffectiveCoordination	m
0	AlAs	152.91209	5.720212	3.110000e-15	5.720212	
1	AlP	140.72993	5.368858	3.150000e-10	5.368858	
2	AlP (F-43m)	162.26119	5.720212	1.330000e-15	5.720212	
3	AlSb	148.62687	5.720212	0.000000e+00	5.720212	
4	B2AsP	142.34014	5.731569	1.021271e-01	5.541343	
...	...	...	...	...	...	
114	TePb	147.79412	6.000000	4.440000e-16	6.000000	
115	Ti2SnC	100.70368	9.383733	1.691867e+00	6.000000	
116	TlBr	160.52725	11.956922	8.880000e-16	11.956922	
117	TlCl	160.26508	11.956922	8.880000e-16	11.956922	
118	TlI	148.12925	11.956922	8.880000e-16	11.956922	

119 rows × 275 columns

示例

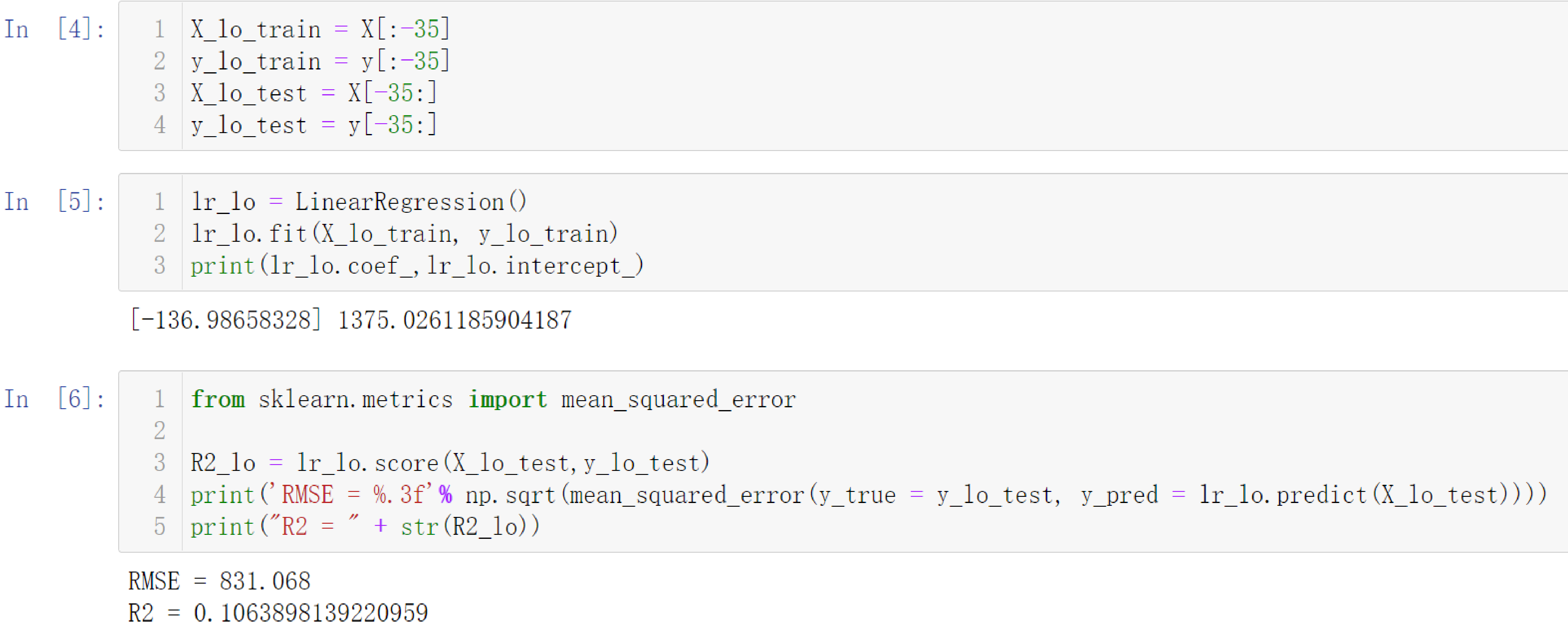


```
In [2]:
X = df['mean_EffectiveCoordination'].values.reshape(-1,1)
y = df['LTC'].values
lr = LinearRegression()
lr.fit(X, y)
plt.figure(figsize = (5, 5))
plt.scatter(df['mean_EffectiveCoordination'], df['LTC'])
plot_x = np.linspace(5, 12)
plot_y = plot_x * lr.coef_ + lr.intercept_
plt.plot(plot_x, plot_y, c = 'r')
plt.show()
```



示例

留出法评价



```
In [4]:
X_lo_train = X[:-35]
y_lo_train = y[:-35]
X_lo_test = X[-35:]
y_lo_test = y[-35:]
```

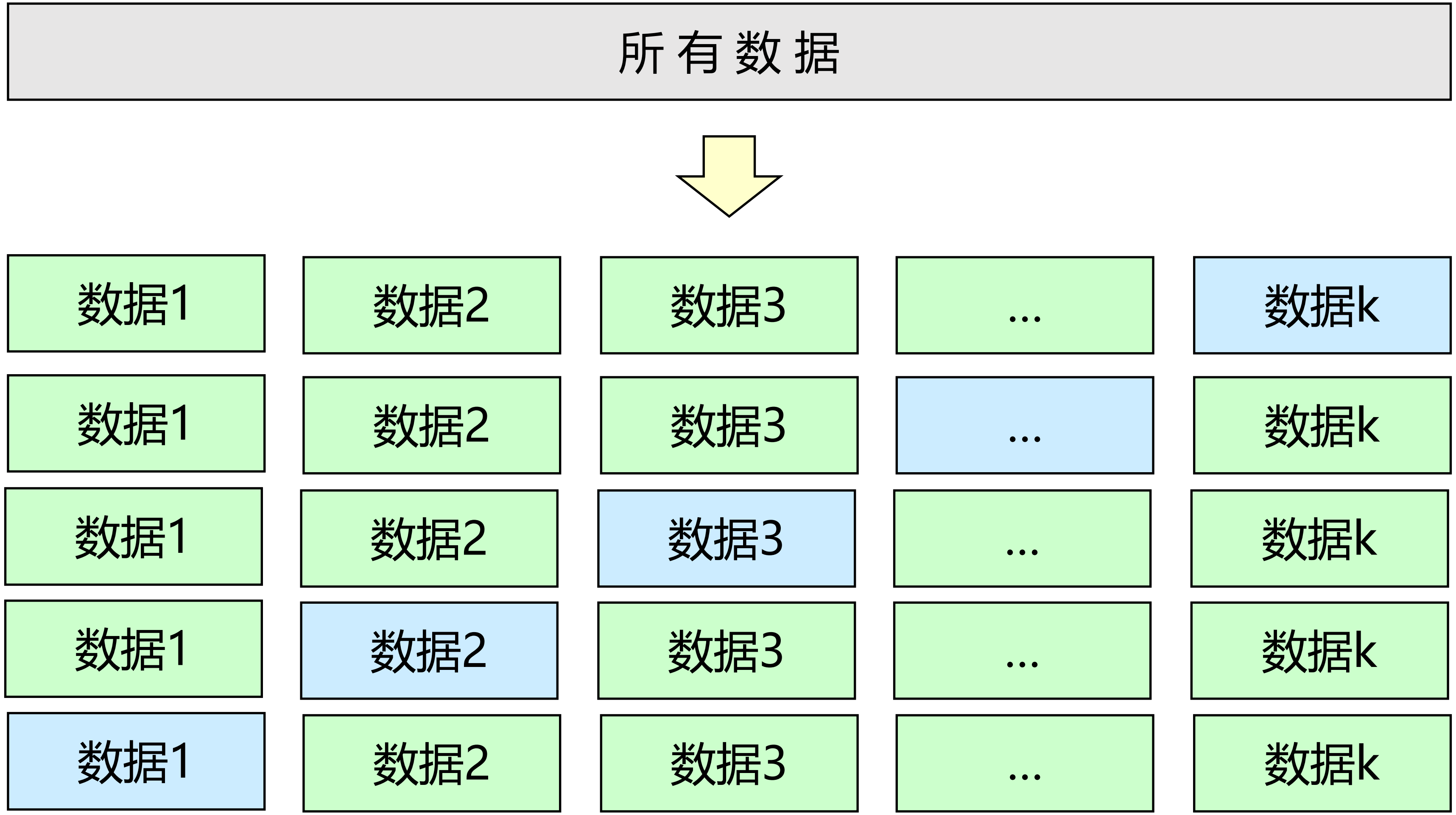
```
In [5]:
lr_lo = LinearRegression()
lr_lo.fit(X_lo_train, y_lo_train)
print(lr_lo.coef_,lr_lo.intercept_)
```

```
[-136.98658328] 1375.0261185904187
```

```
In [6]:
from sklearn.metrics import mean_squared_error

R2_lo = lr_lo.score(X_lo_test,y_lo_test)
print(’RMSE = %.3f’% np.sqrt(mean_squared_error(y_true = y_lo_test, y_pred = lr_lo.predict(X_lo_test))))
print(”R2 = ” + str(R2_lo))
```

```
RMSE = 831.068
R2 = 0.1063898139220959
```

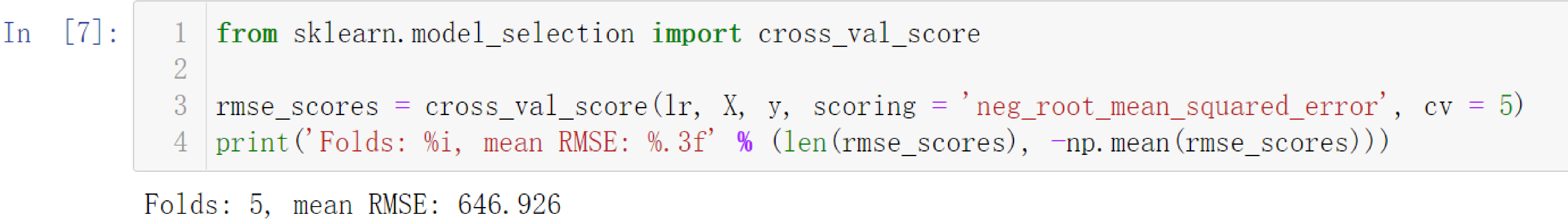
交叉验证法（Cross Validation）



所有数据

🟩 数据1	🟩 数据2	🟩 数据3	🟩 …	🟦 数据k
🟩 数据1	🟩 数据2	🟩 数据3	🟦 …	🟩 数据k
🟩 数据1	🟩 数据2	🟦 数据3	🟩 …	🟩 数据k
🟩 数据1	🟦 数据2	🟩 数据3	🟩 …	🟩 数据k
🟦 数据1	🟩 数据2	🟩 数据3	🟩 …	🟩 数据k

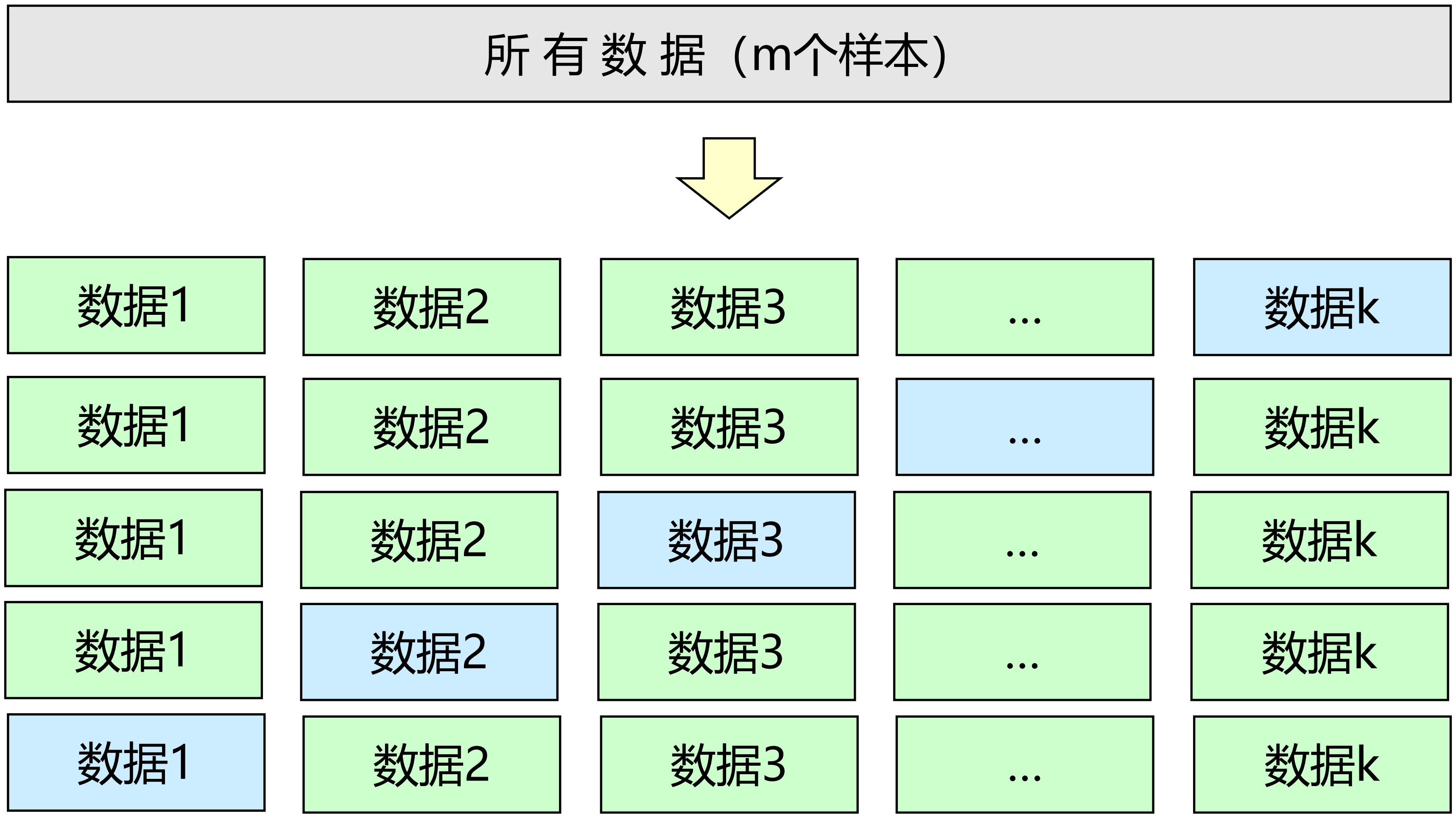
样例：交叉验证



```
In [7]:
from sklearn.model_selection import cross_val_score

rmse_scores = cross_val_score(lr, X, y, scoring = 'neg_root_mean_squared_error', cv = 5)
print('Folds: %i, mean RMSE: %.3f' % (len(rmse_scores), -np.mean(rmse_scores)))
```

```
Folds: 5, mean RMSE: 646.926
```

留一法（Leave-One-Out）



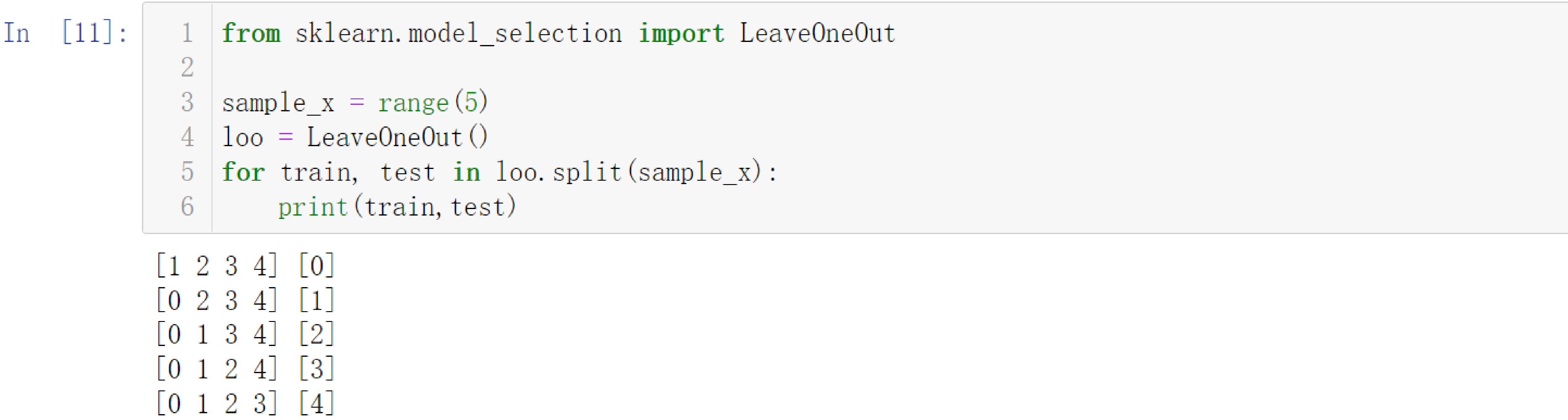
所 有 数 据 （m个样本）

数据1	数据2	数据3	…	数据k
数据1	数据2	数据3	…	数据k
数据1	数据2	数据3	…	数据k
数据1	数据2	数据3	…	数据k
数据1	数据2	数据3	…	数据k

k = m

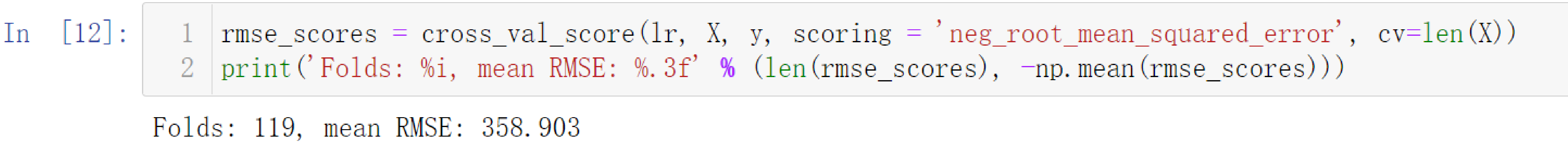
每次只拿一个样本做测试

样例：留一法的使用



```
In [11]:
from sklearn.model_selection import LeaveOneOut

sample_x = range(5)
loo = LeaveOneOut()
for train, test in loo.split(sample_x):
    print(train,test)
```

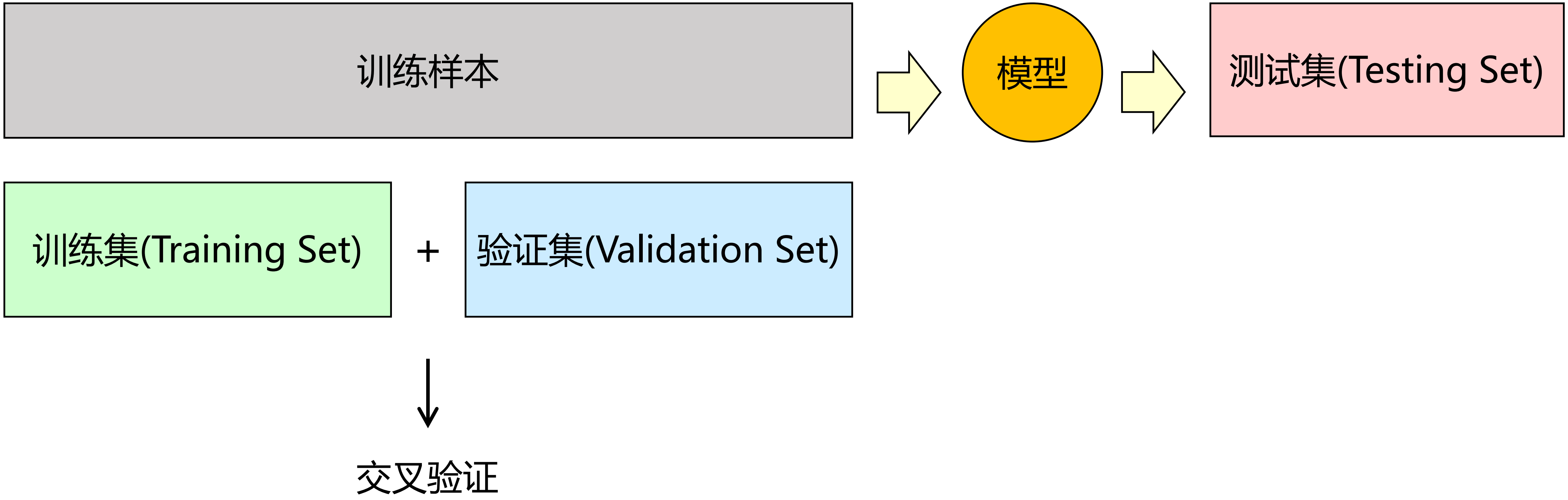
```
[1 2 3 4] [0]
[0 2 3 4] [1]
[0 1 3 4] [2]
[0 1 2 4] [3]
[0 1 2 3] [4]
```



```
In [12]:
rmse_scores = cross_val_score(lr, X, y, scoring = 'neg_root_mean_squared_error’, cv=len(X))
print('Folds: %i, mean RMSE: %.3f’ % (len(rmse_scores), -np.mean(rmse_scores)))
```

```
Folds: 119, mean RMSE: 358.903
```

交叉验证法（Cross Validation)



训练样本

模型

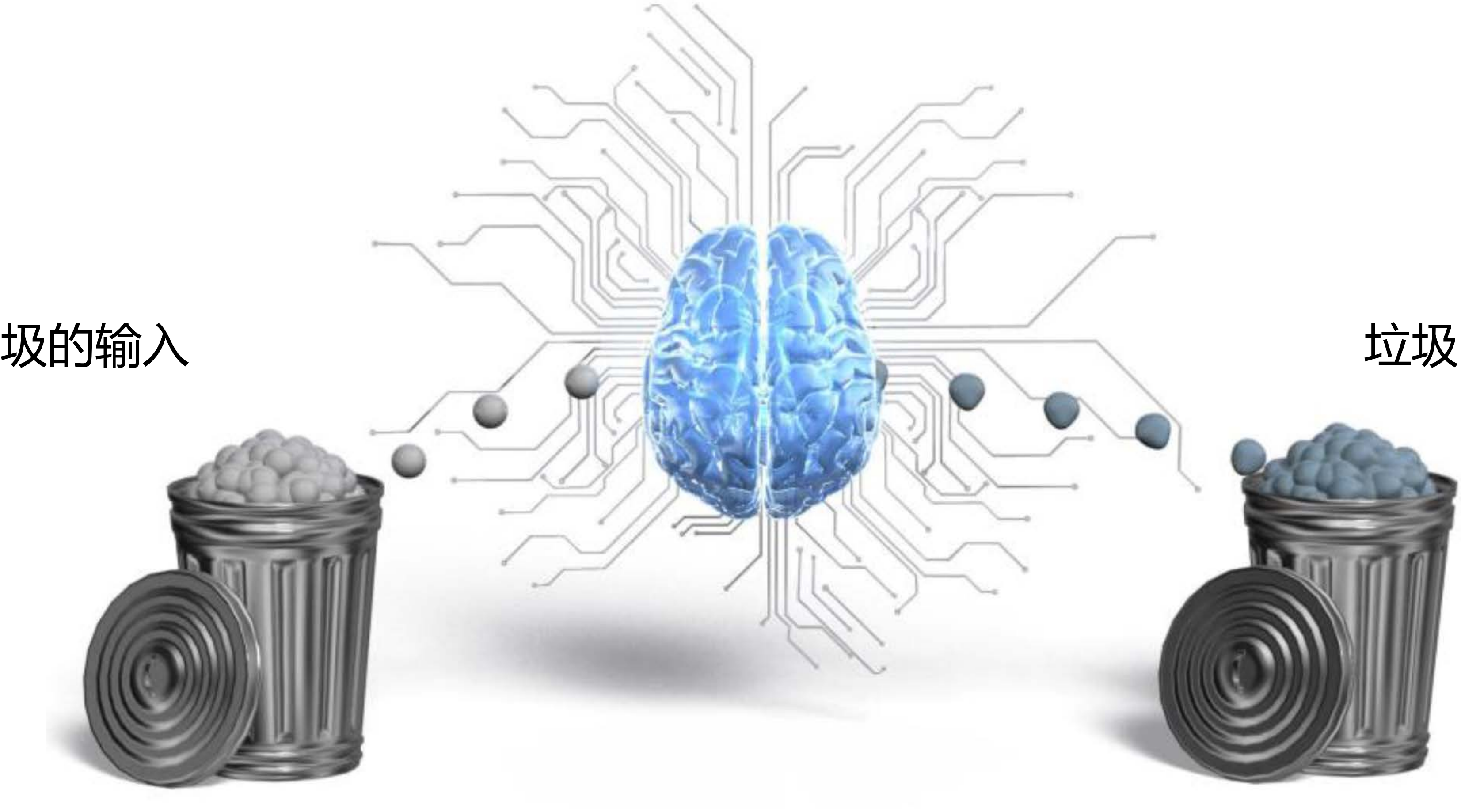
测试集(Testing Set)

训练集(Training Set) + 验证集(Validation Set)

交叉验证

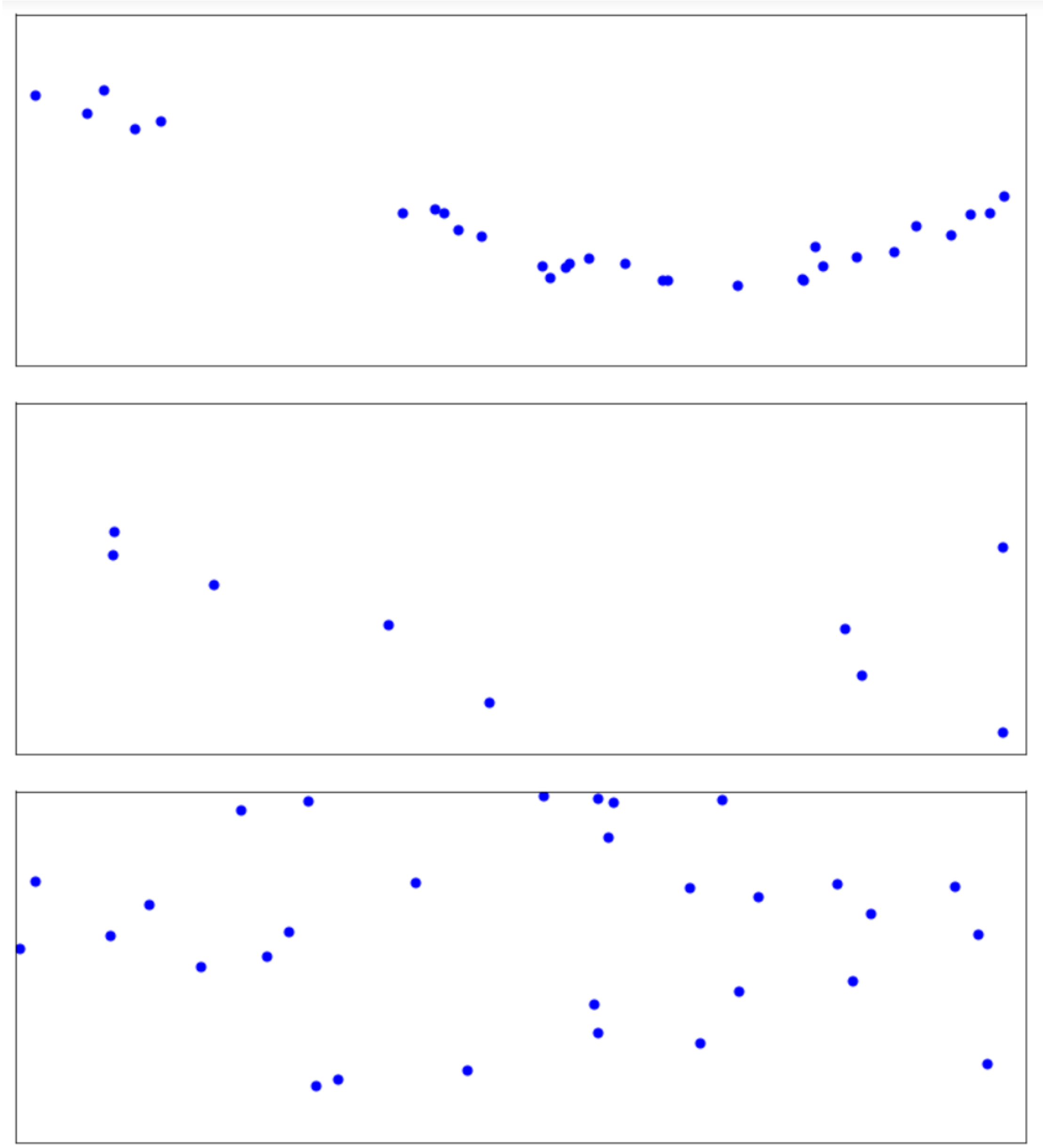
Garbage in, garbage out

永远记住，你**输入**数据的质量决定了你**输出**数据集的质量



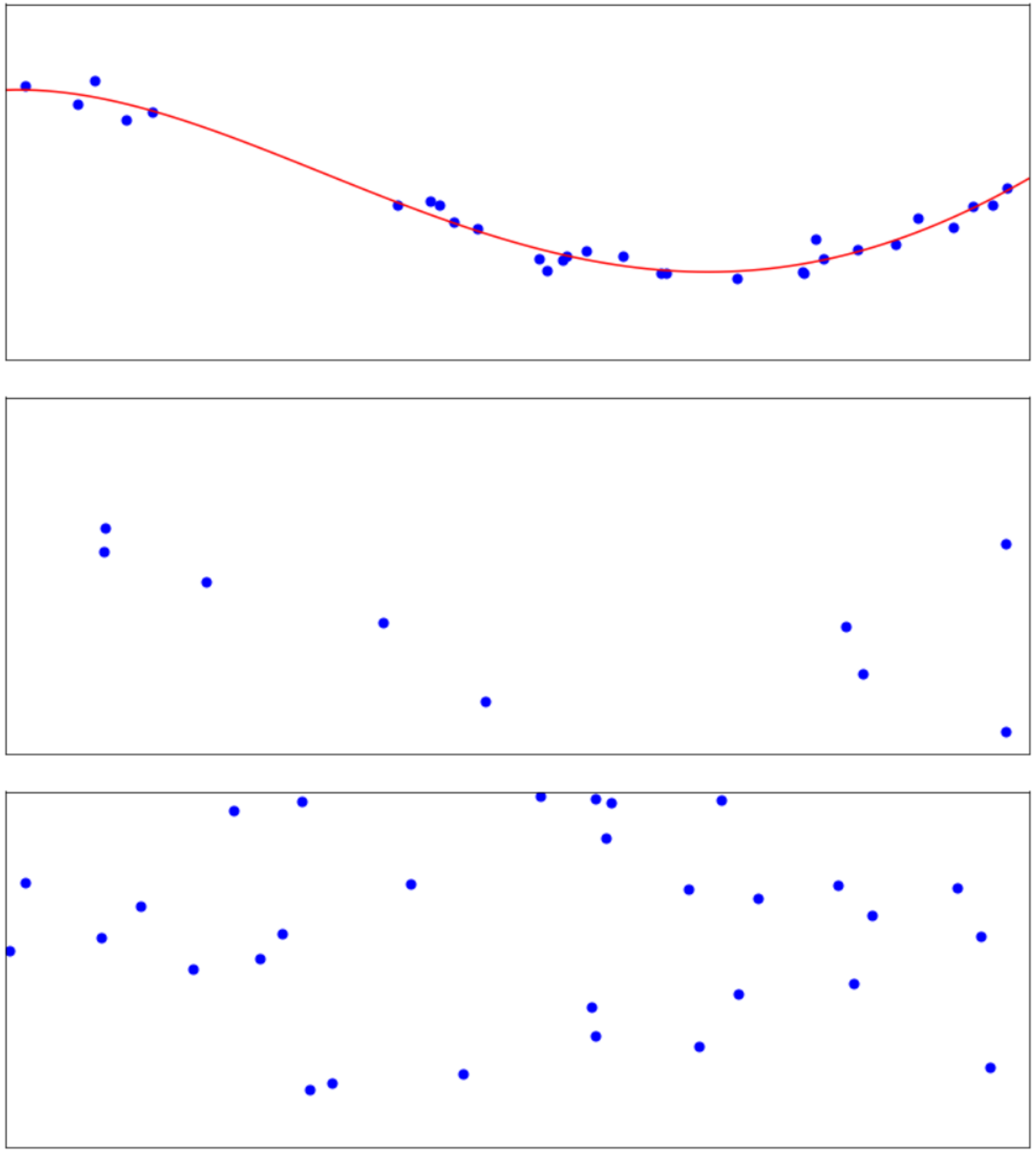
T. Toyao, K. Shimizu, *et. al. ACS Catal.* **2020**, 10, 2260.

Garbage in, garbage out



对这三组数据分别进行回归

Garbage in, garbage out

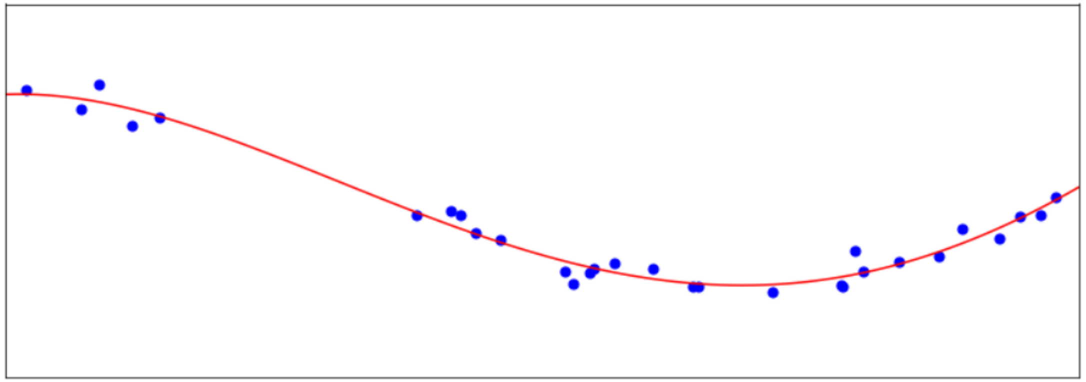


回归曲线基本符合数据点趋势

效果很棒

充足的有规律的数据

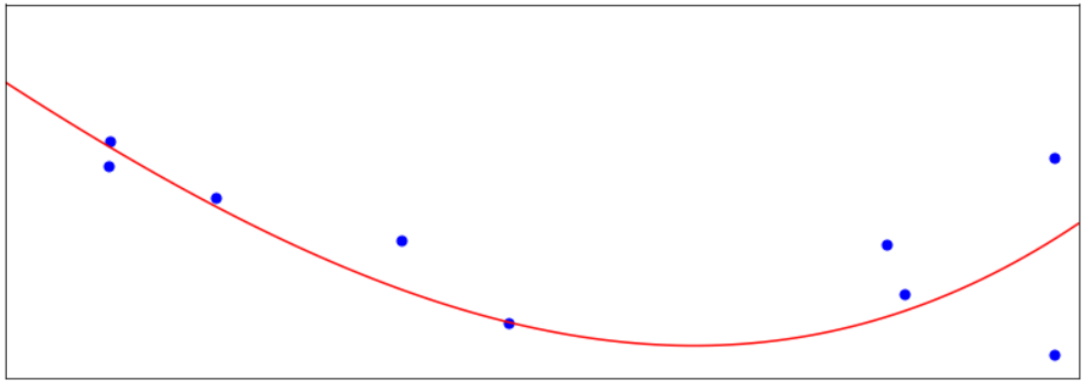
Garbage in, garbage out



回归曲线基本符合数据点趋势

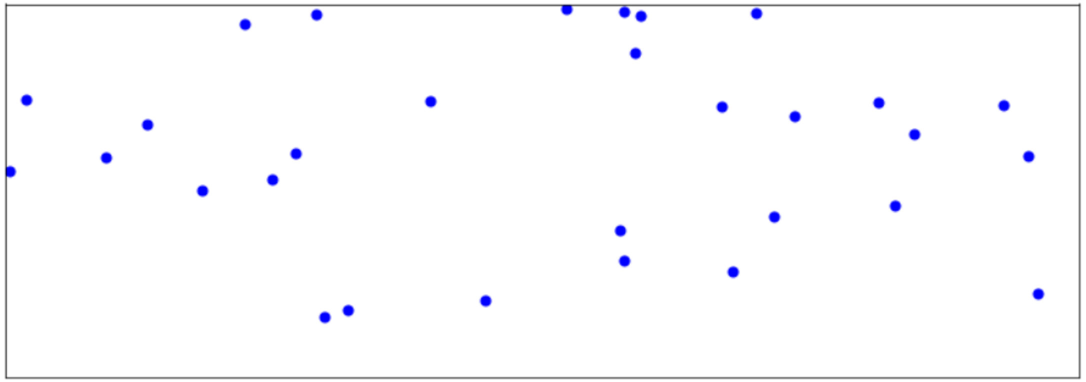
效果很棒

充足的有规律的数据

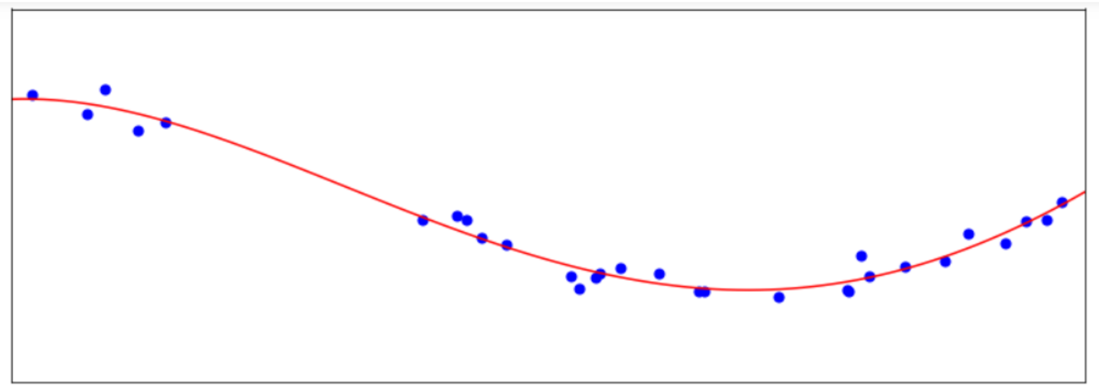


效果还不好说

数据不充足



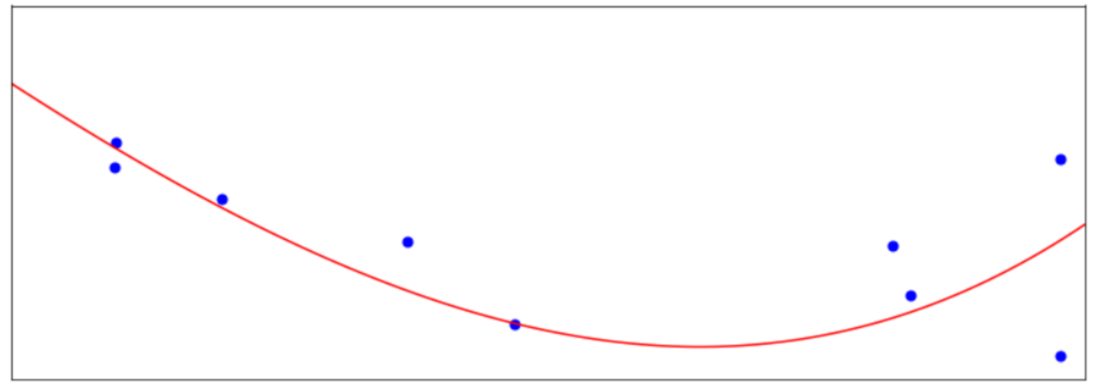
Garbage in, garbage out



回归曲线基本符合数据点趋势

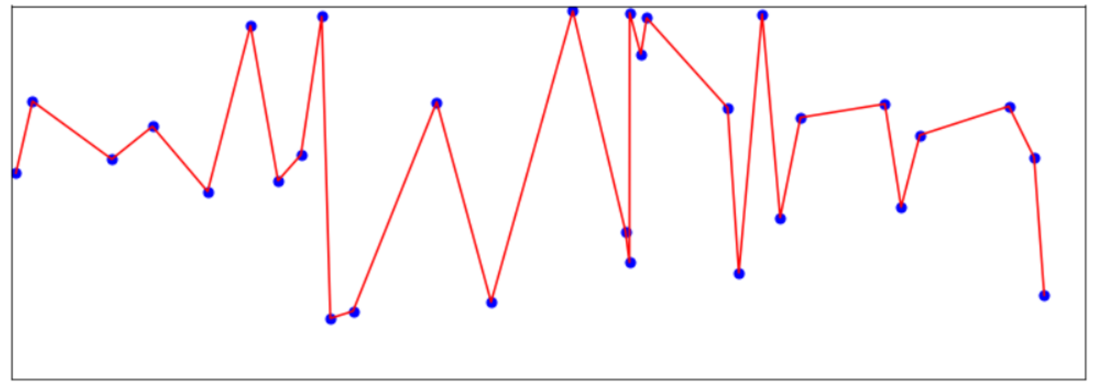
效果很棒

充足的有规律的数据



效果还不好说

数据不充足

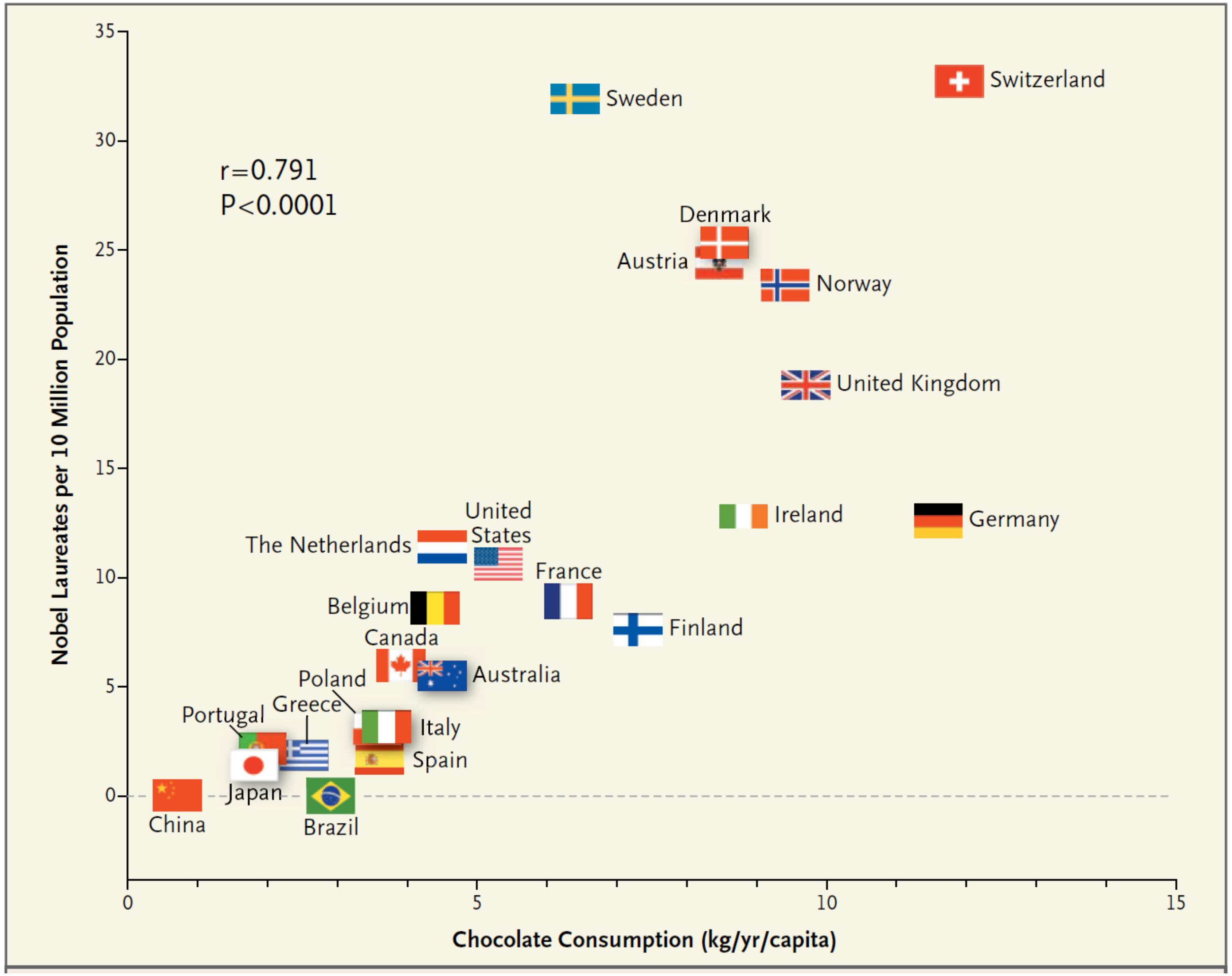


回归曲线完全不可用

输入与输出之间完全没有相关性

充足的“垃圾“数据

无关数据



巧克力消费量与获得诺贝尔奖人数的关系

N. Engl. J. Med. 可不是“野鸡”杂志，而是医学界声望非常高的杂志。

影响因子

2019-2020

74.699

5.7%

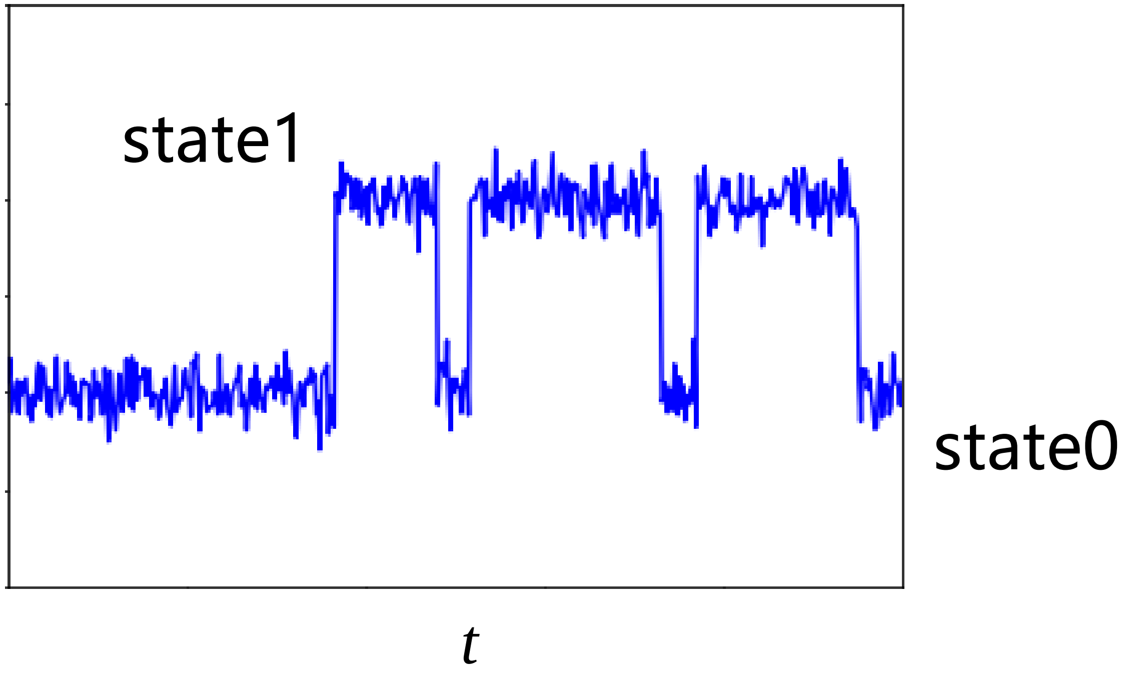
H. Franz and M. D. Messerli, *N. Engl. J. Med.* **2012**, 367, 1562.

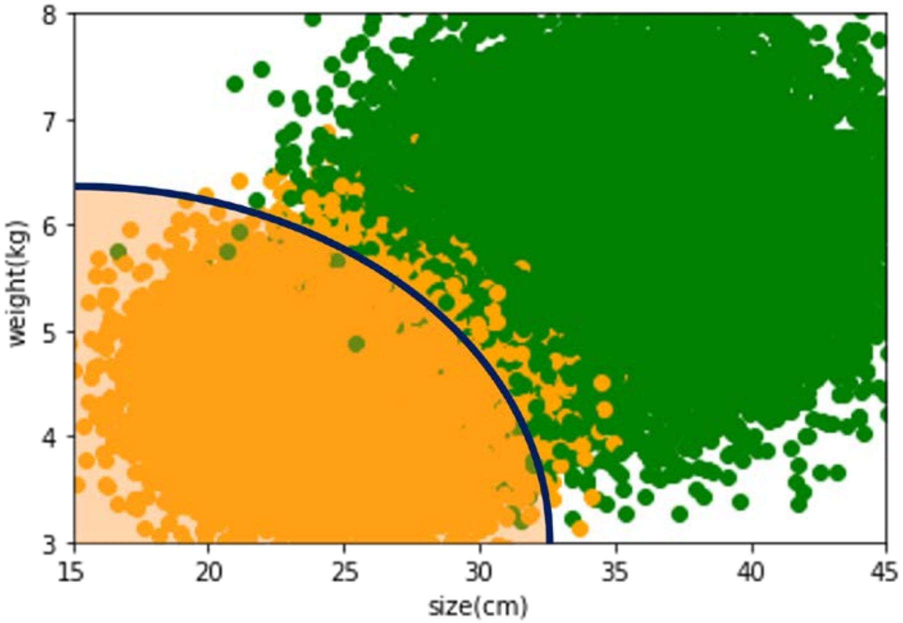
目录

1. 算法简介
2. 回归算法
3. 双金属吸附中的回归
4. 非线性回归求速率常数
5. 模型评价
6. 决策树分类算法
7. 支持向量机

分类算法

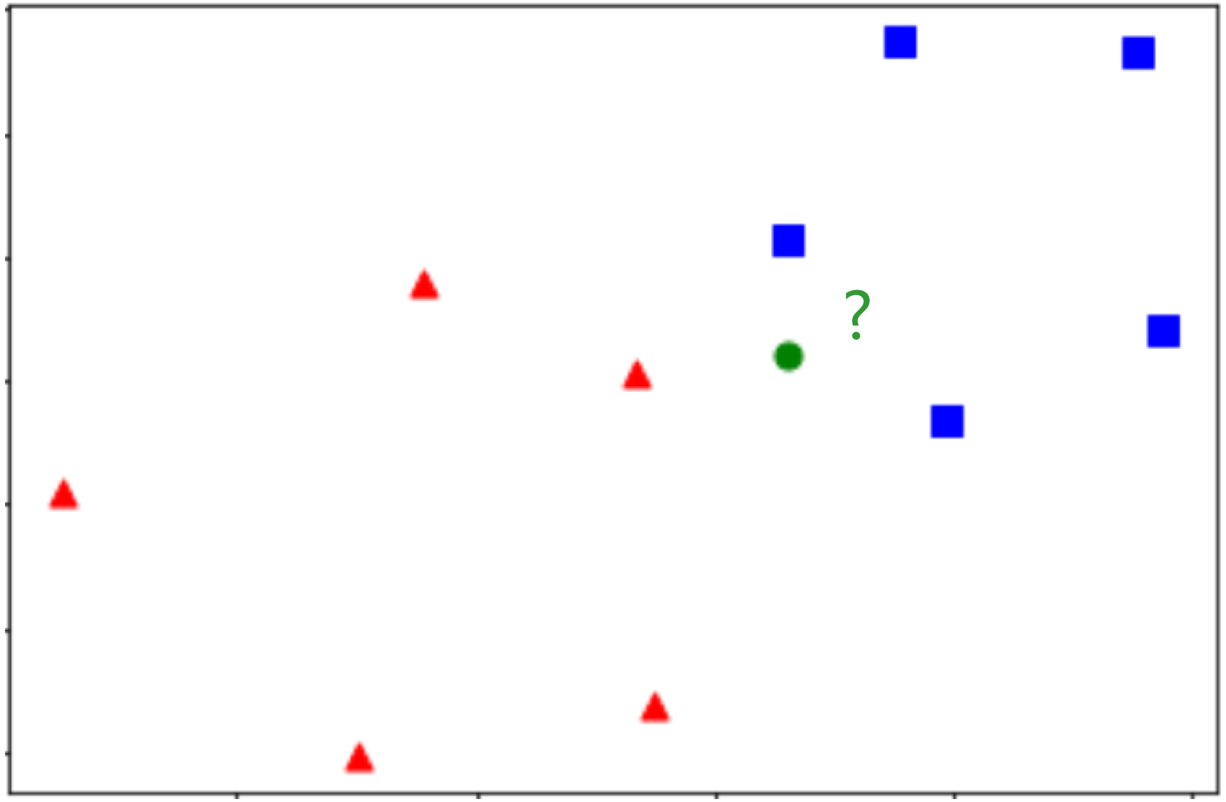
在一个已知标号的样本中找到一种分类器，使其能对未知样品进行分类。





分类问题

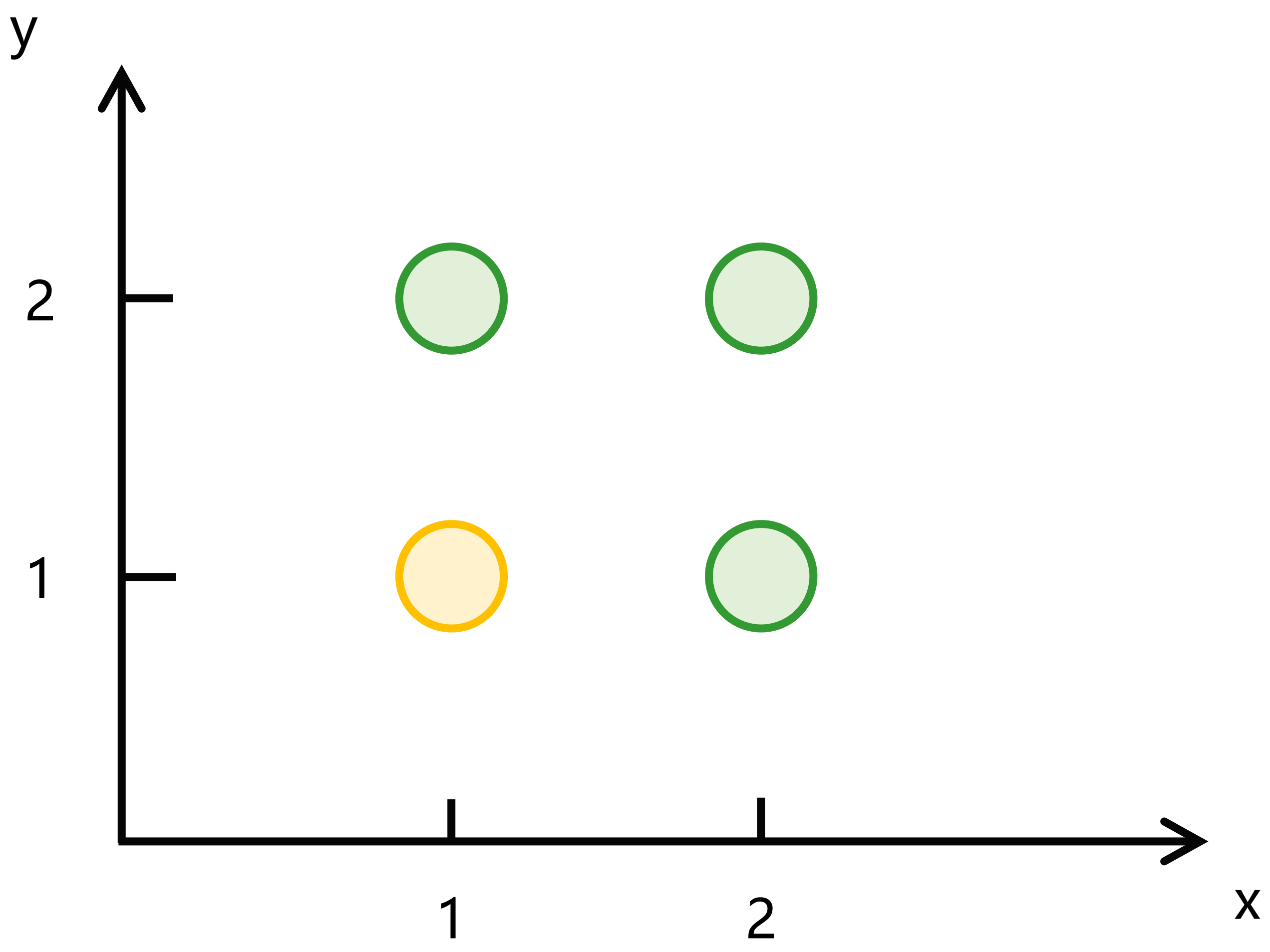
分类问题：已知红色点为A类，蓝色点为B类，绿色点应该属于哪一类?



十大经典算法

排名	算法	作者	发表年份
1	C4.5	Quinlan, J. R.	1993
2	k-Means	MacQueen, J. B.	1967
3	SVM	Vapnik, V. N.	1995
4	Apriori	Agrawal, R.	1994
5	EM	McLachlan, G.	2000
6	PageRank	Brin, S.	1998
7	AdaBoost	Freund, Y.	1997
8	kNN	Hastie, T.	1996
9	Naïve Bayes	Hand, D. J.	2001
10	CART	Breiman, L.	1984

决策树算法

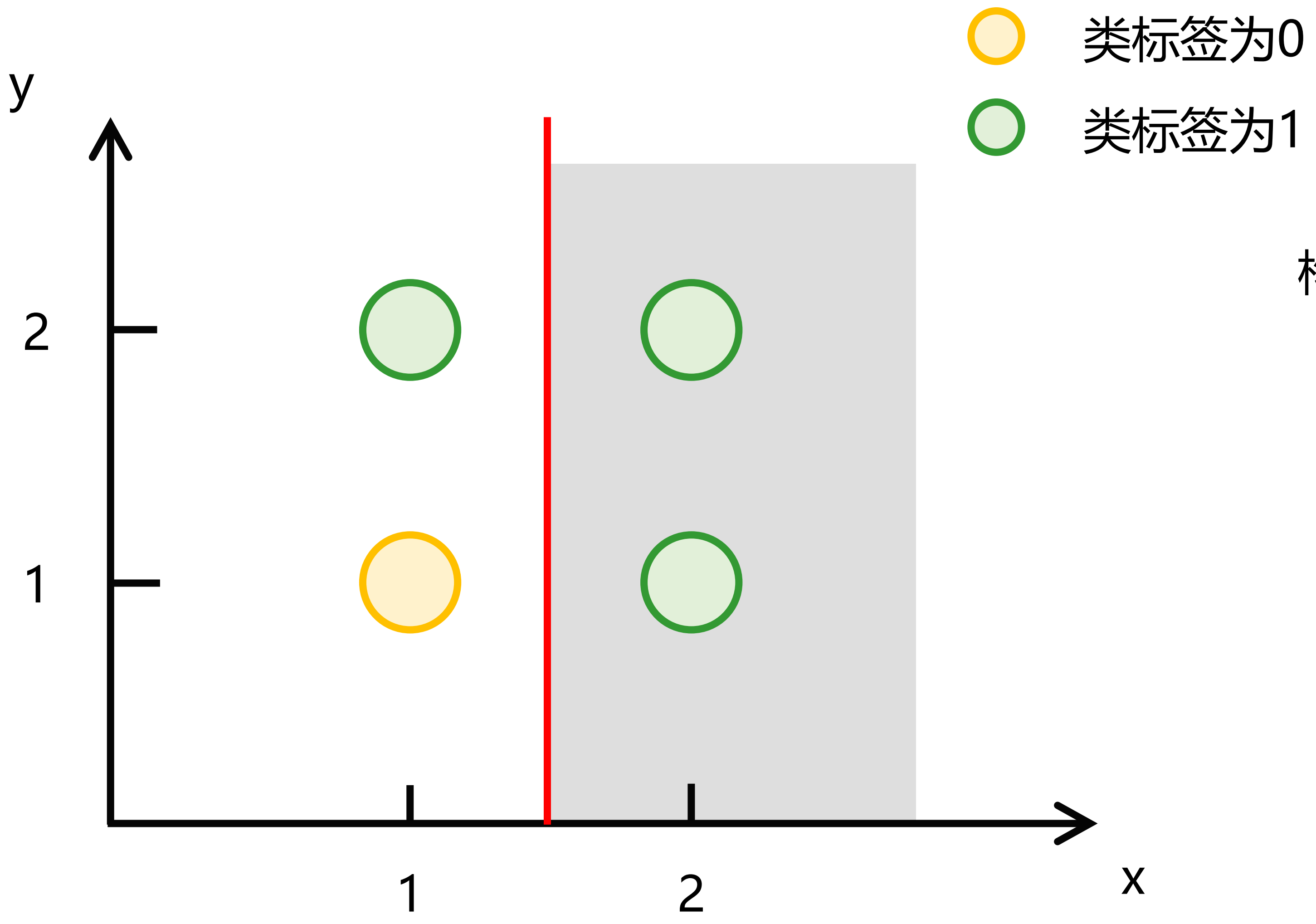


🟡 类标签为0

🟢 类标签为1

已知新数据的x与y值，
如何对其进行分类？

决策树算法

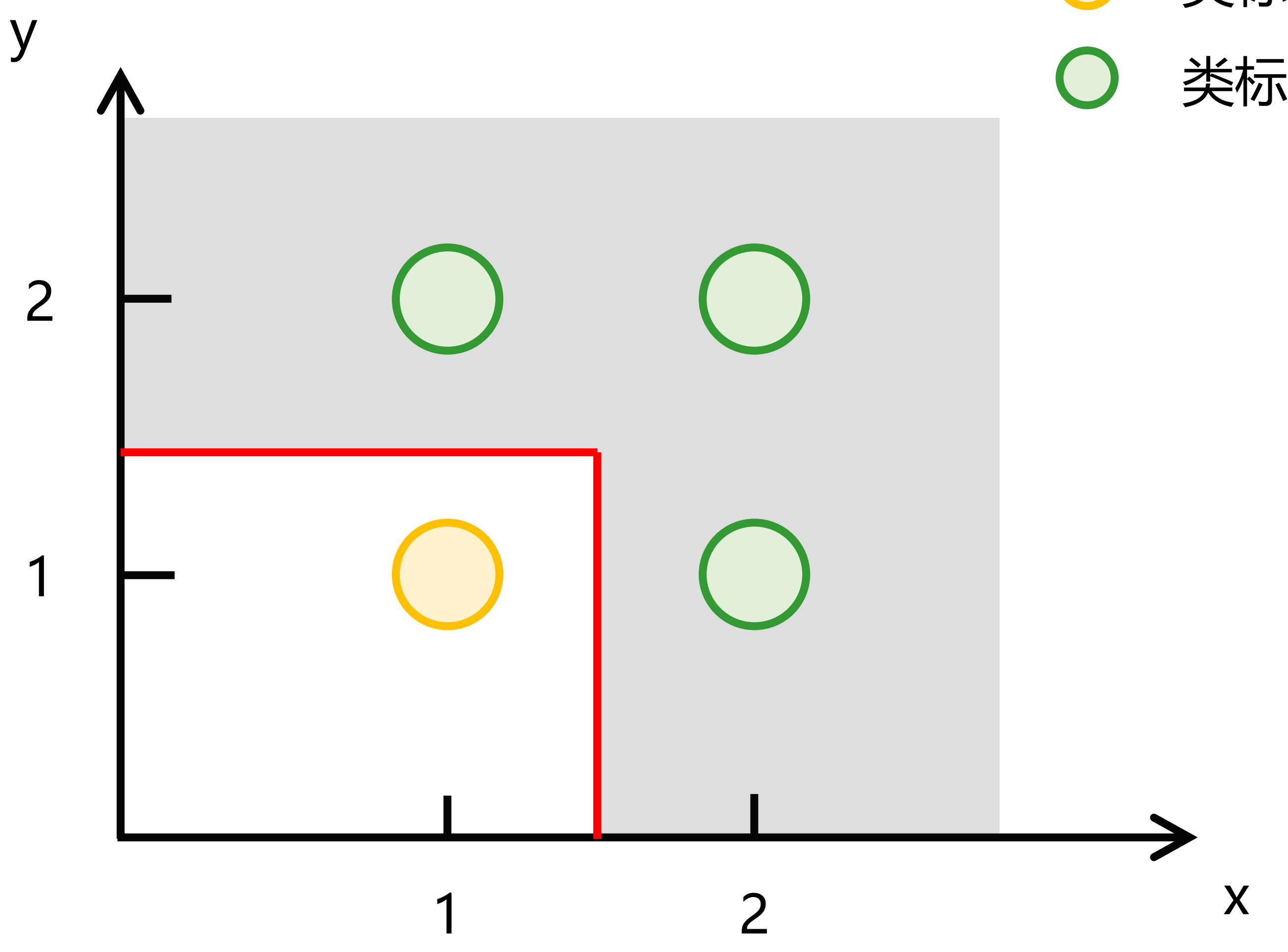


已知新数据的x与y值，如何对其进行分类？

构建模型：

1. 先按x数据进行分类

决策树算法



○ 类标签为0

○ 类标签为1

已知新数据的x与y值，
如何对其进行分类?

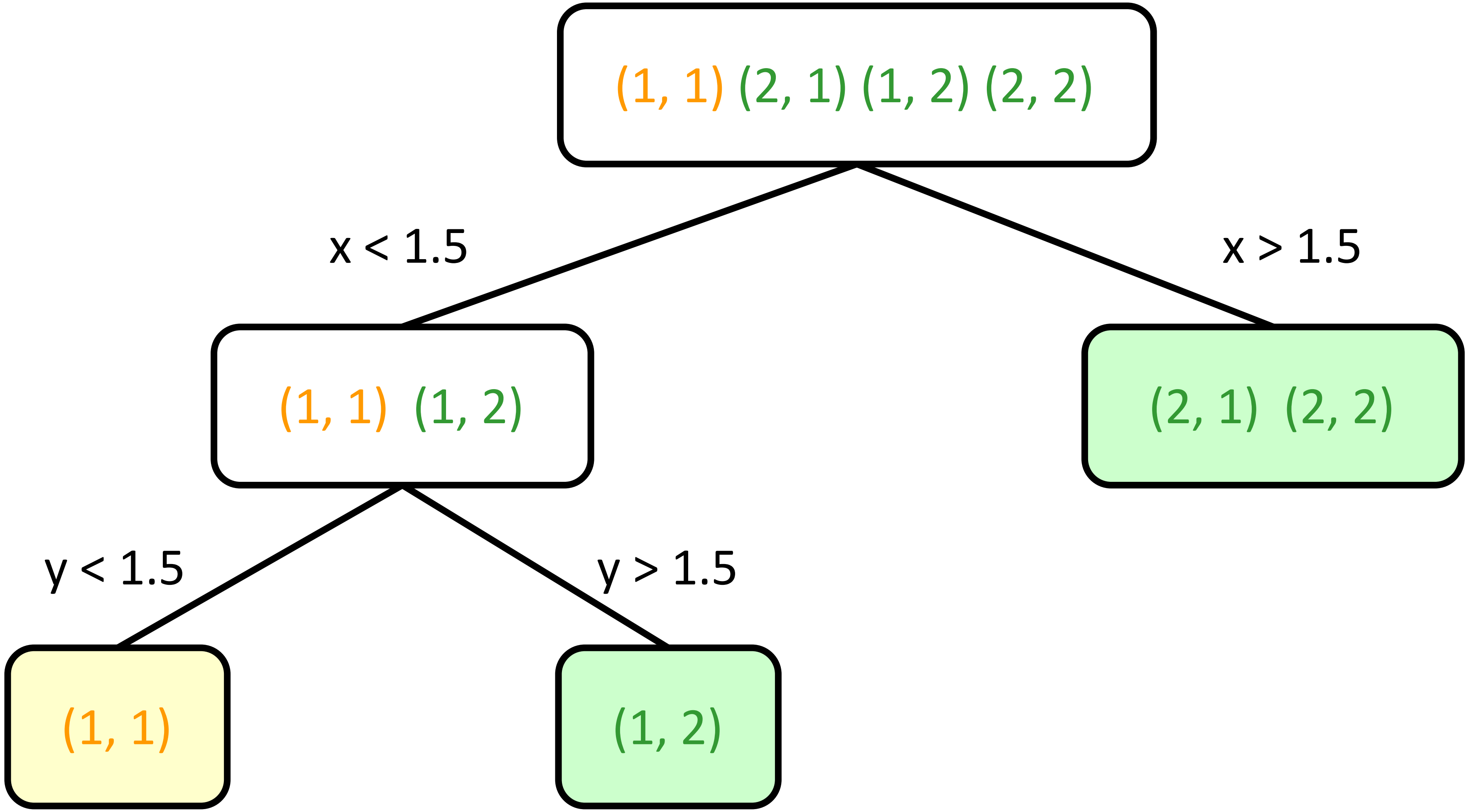
构建模型：

1. 先按x数据进行分类

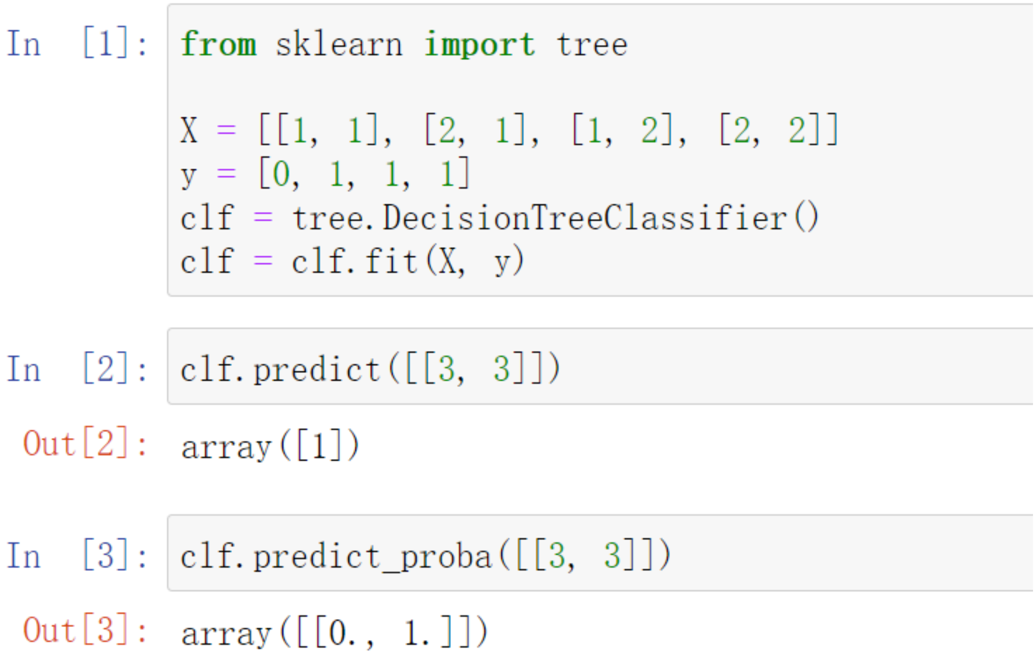
2. 对未分类点再按y数据进行分类

对未知点按模型结果进行分类

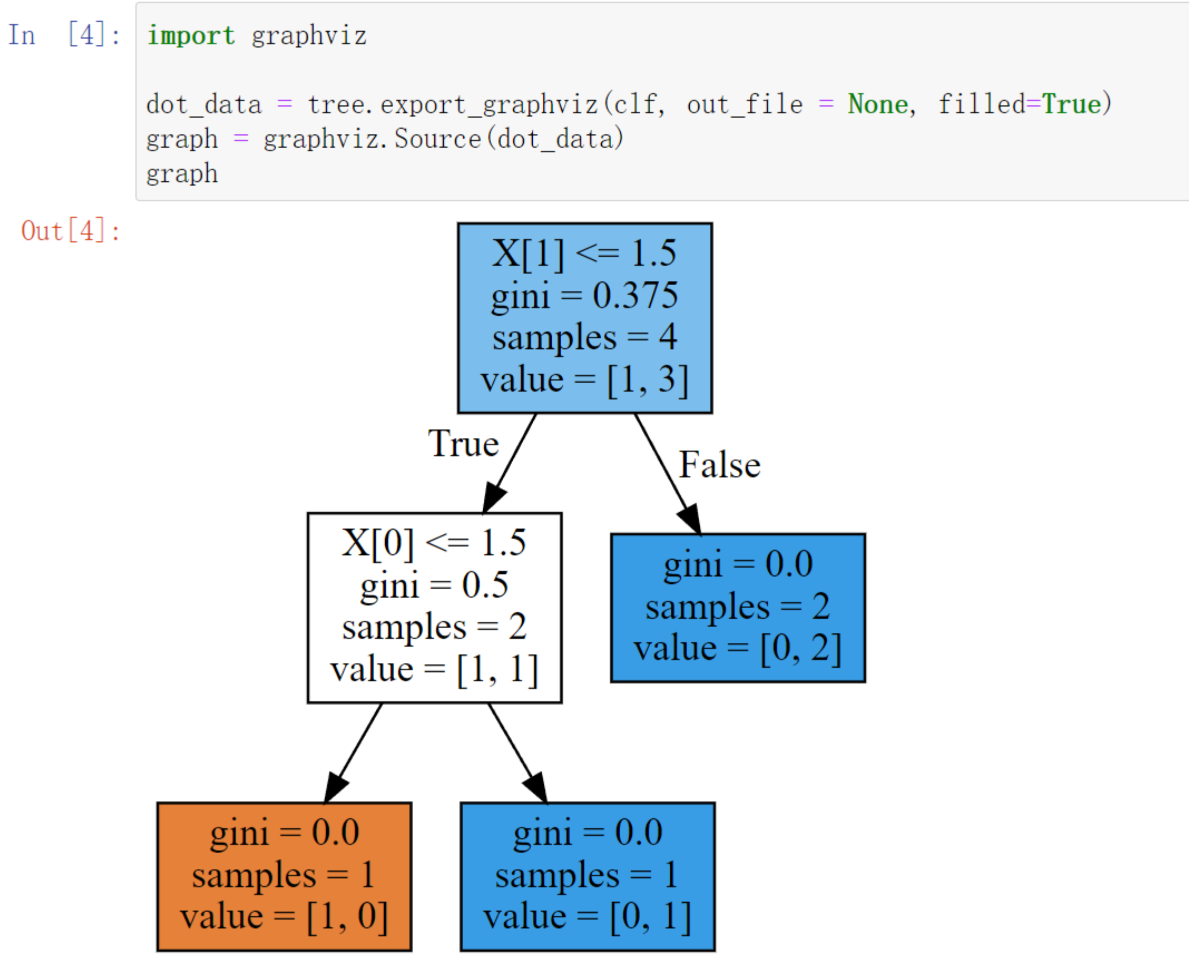
决策树算法



决策树算法



```
In [1]: from sklearn import tree

        X = [[1, 1], [2, 1], [1, 2], [2, 2]]
        y = [0, 1, 1, 1]
        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(X, y)

In [2]: clf.predict([[3, 3]])

Out[2]: array([1])

In [3]: clf.predict_proba([[3, 3]])

Out[3]: array([[0., 1.]])
```

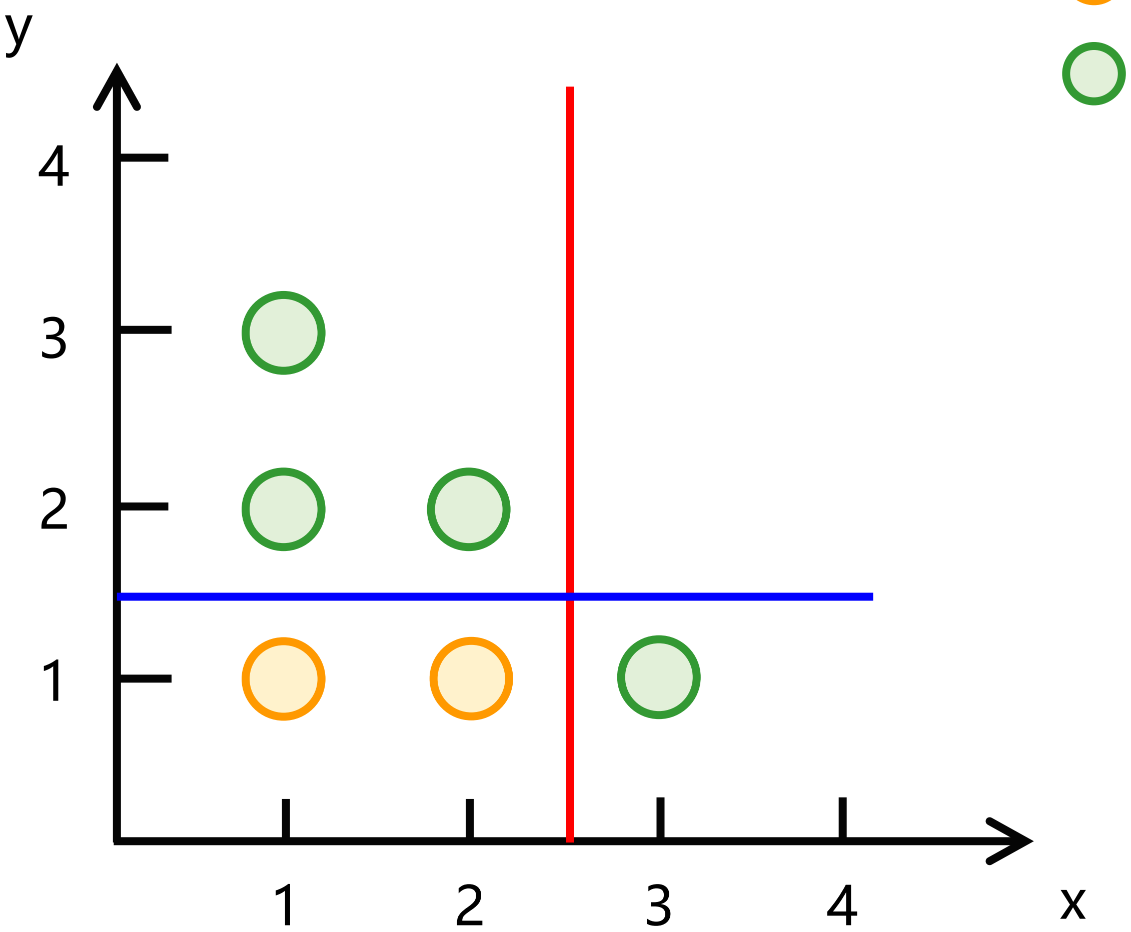


```
In [4]: import graphviz

        dot_data = tree.export_graphviz(clf, out_file = None, filled=True)
        graph = graphviz.Source(dot_data)
        graph

Out[4]:
```

划分选择

🟠 类标签为0

🟢 类标签为1



先按红线还是蓝线分类?

信息熵

信息熵（information entropy）：

假定当前样本集合D中第k类样本所占的比例为p_k,则D的信息熵为

$$Ent(D) = -\sum p_k log_2 p_k$$

Ent(D)的值越小，D的纯度越高

假定共含有n种样本，最大熵对应的分布为

$$p_k = \frac{1}{n}$$

ID3算法

信息增益（information gain）：

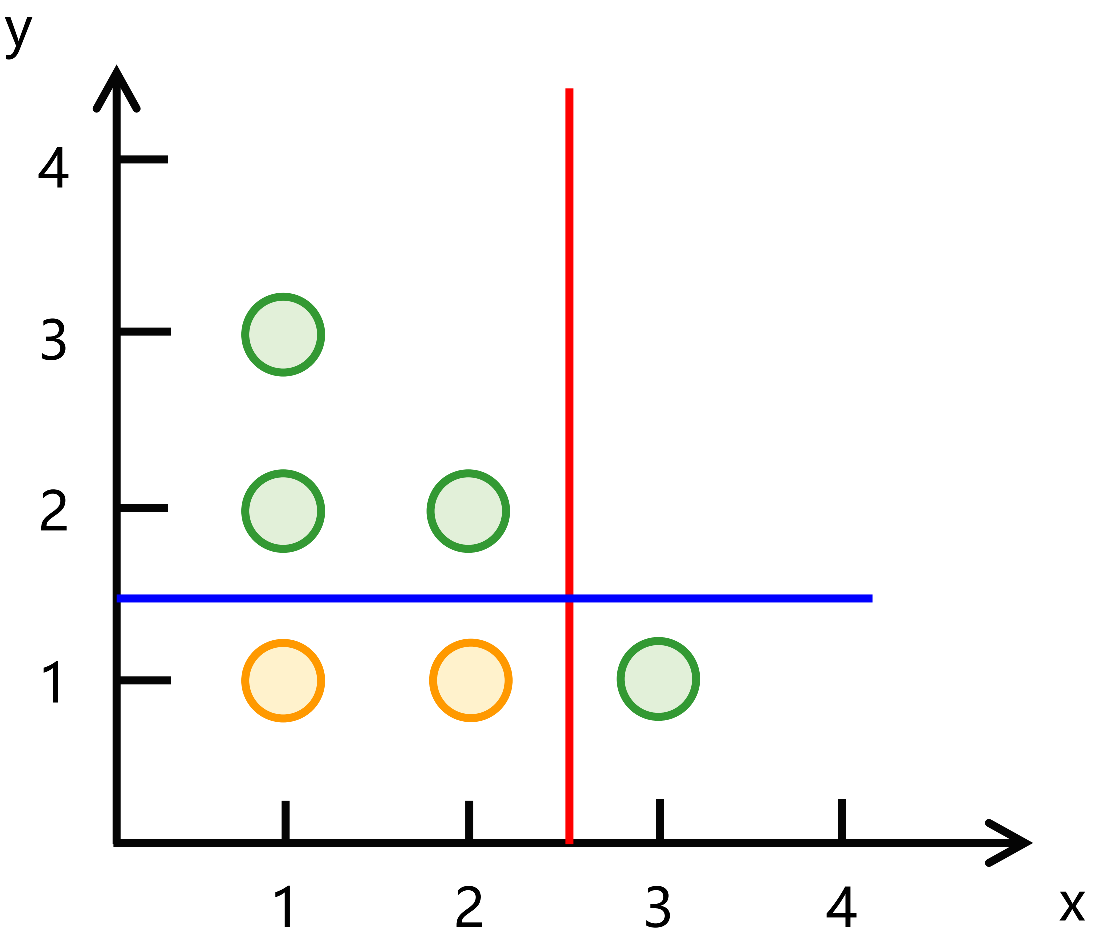
划分后样本的纯度提升

$$Gain(D,a) = Ent(D) - \sum \frac{|D^v|}{|D|} Ent(D^v)$$

ID3学习算法(Quinlan, 1975):

$$\boxed{a_* = \underset{a \in A}{argmax}\, Gain(D,a)}$$

划分选择



划分前：

$$Ent(D) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$

红线划分后：

$$Ent(D^1) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.9709$$

$$Ent(D^2) = 0$$

$$Gain(D,a) = 0.9183 - \frac{5}{6} \times 0.9709 = 0.1092$$

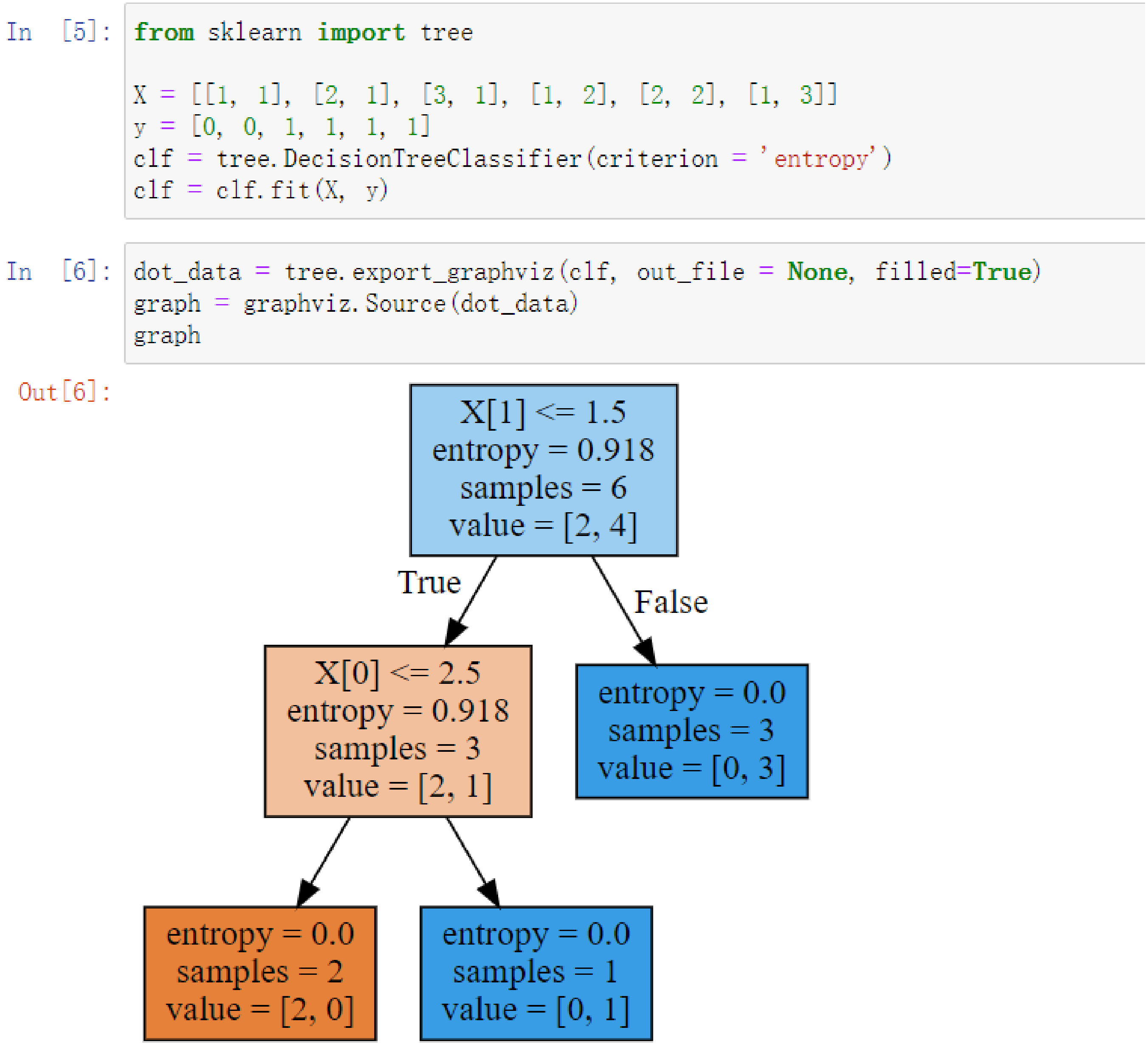
蓝线划分后：

$$Ent(D^1) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$

$$Ent(D^2) = 0$$

$$Gain(D,a) = 0.9183 - \frac{1}{2} \times 0.9183 = 0.4592$$

划分选择



In [5]:

```
from sklearn import tree

X = [[1, 1], [2, 1], [3, 1], [1, 2], [2, 2], [1, 3]]
y = [0, 0, 1, 1, 1, 1]
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
```

In [6]:

```
dot_data = tree.export_graphviz(clf, out_file = None, filled=True)
graph = graphviz.Source(dot_data)
graph
```

Out[6]:

C4.5算法

C4.5算法(Quinlan, 1993)

使用增益率进行划分

$$Gain\_ratio(D,a)=\frac{Gain(D,a)}{IV(a)}$$

其中

$$IV(a)=-\sum\frac{|D^v|}{|D|}log_2\frac{|D^v|}{|D|}$$

CART算法

基尼值

$$Gini(D) = \sum_{k} \sum_{k\prime \neq k} p_k p_{k\prime}$$

$$= 1 - \sum_{k} p_k^2$$

属性a的基尼指数

$$Gini\_index(D, a) = \sum \frac{|D^v|}{|D|} Gini(D^v)$$

CART算法(Breiman, 1984)

$$\boxed{a_* = \underset{a \in A}{argmin}\, Gini\_index(D, a)}$$

分类背景：有机小分子

定量构效关系(quantitative structure-activity relationships, QSAR)

药物、农药、化学毒剂等研究

麻醉化合物小分子通常被分为两类，他们的毒性机理有较大的区别

第一类：非极性麻醉剂

第二类：极性麻醉剂

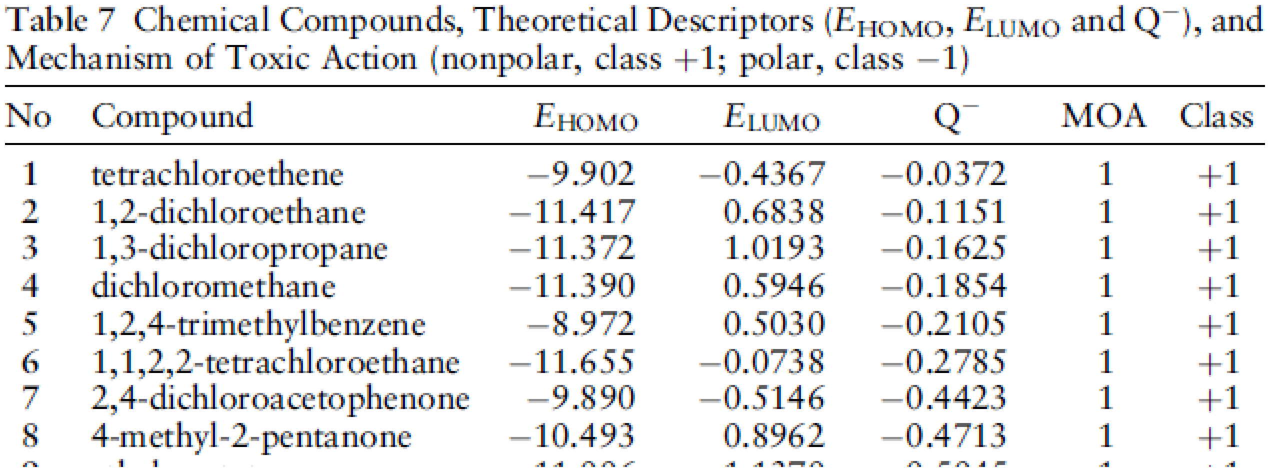
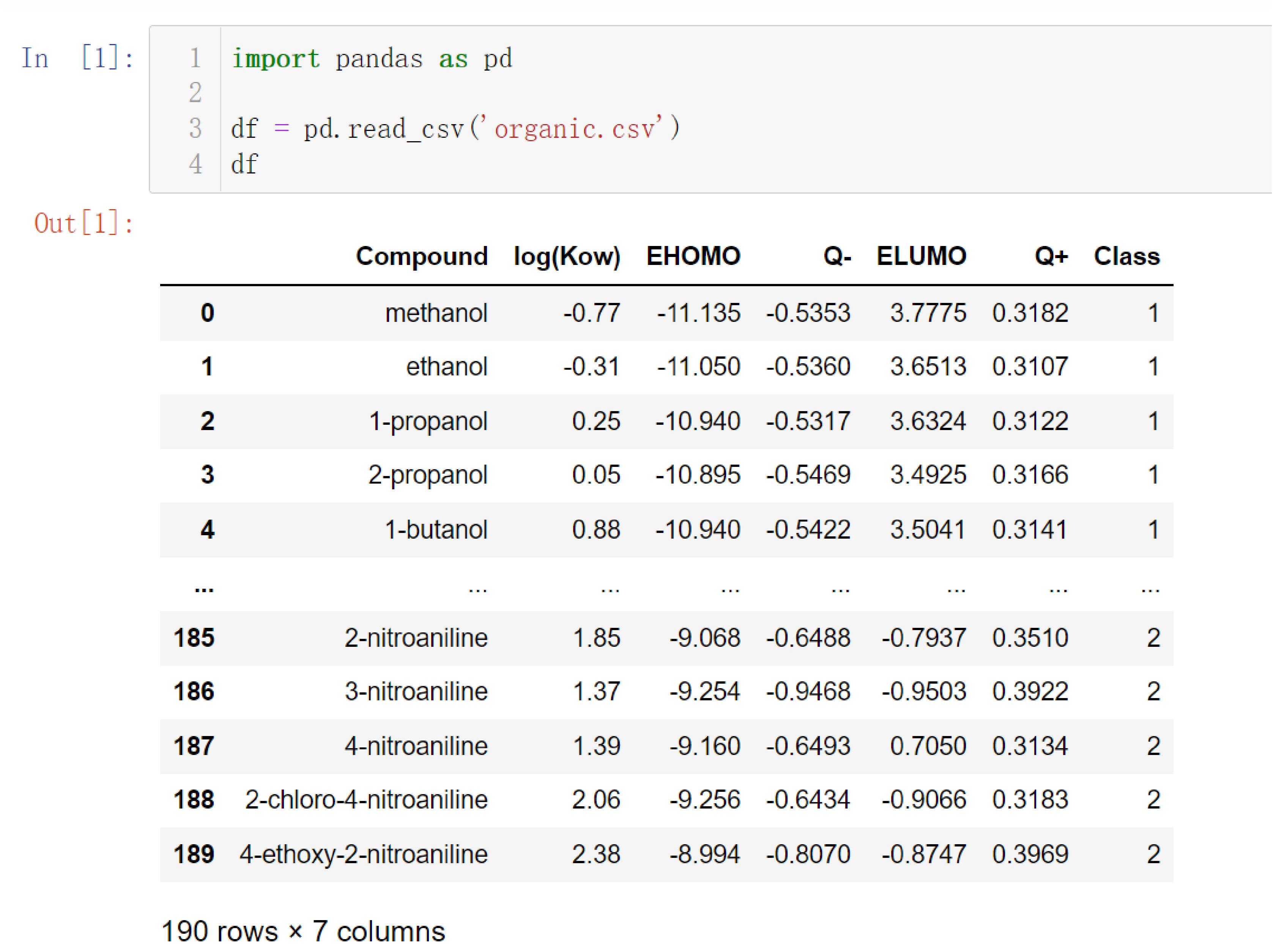


Table 7 Chemical Compounds, Theoretical Descriptors (E_{HOMO}, E_{LUMO} and Q^-), and Mechanism of Toxic Action (nonpolar, class +1; polar, class −1)

No	Compound	E_{HOMO}	E_{LUMO}	Q^-	MOA	Class
1	tetrachloroethene	−9.902	−0.4367	−0.0372	1	+1
2	1,2-dichloroethane	−11.417	0.6838	−0.1151	1	+1
3	1,3-dichloropropane	−11.372	1.0193	−0.1625	1	+1
4	dichloromethane	−11.390	0.5946	−0.1854	1	+1
5	1,2,4-trimethylbenzene	−8.972	0.5030	−0.2105	1	+1
6	1,1,2,2-tetrachloroethane	−11.655	−0.0738	−0.2785	1	+1
7	2,4-dichloroacetophenone	−9.890	−0.5146	−0.4423	1	+1
8	4-methyl-2-pentanone	−10.493	0.8962	−0.4713	1	+1

决策树分类



In [1]:

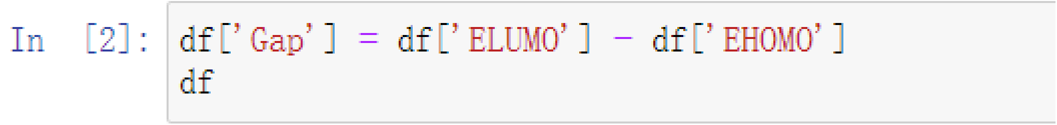
```
import pandas as pd

df = pd.read_csv(’organic.csv’)
df
```

Out[1]:

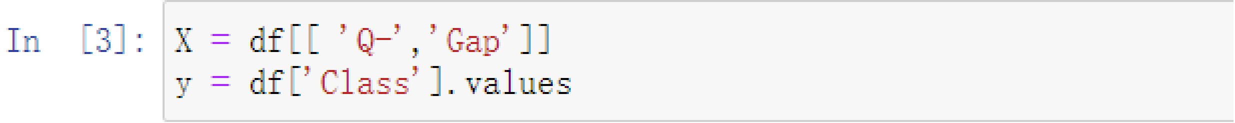
	Compound	log(Kow)	EHOMO	Q-	ELUMO	Q+	Class
0	methanol	-0.77	-11.135	-0.5353	3.7775	0.3182	1
1	ethanol	-0.31	-11.050	-0.5360	3.6513	0.3107	1
2	1-propanol	0.25	-10.940	-0.5317	3.6324	0.3122	1
3	2-propanol	0.05	-10.895	-0.5469	3.4925	0.3166	1
4	1-butanol	0.88	-10.940	-0.5422	3.5041	0.3141	1
...	...	...	...	...	...	...	...
185	2-nitroaniline	1.85	-9.068	-0.6488	-0.7937	0.3510	2
186	3-nitroaniline	1.37	-9.254	-0.9468	-0.9503	0.3922	2
187	4-nitroaniline	1.39	-9.160	-0.6493	0.7050	0.3134	2
188	2-chloro-4-nitroaniline	2.06	-9.256	-0.6434	-0.9066	0.3183	2
189	4-ethoxy-2-nitroaniline	2.38	-8.994	-0.8070	-0.8747	0.3969	2

190 rows × 7 columns

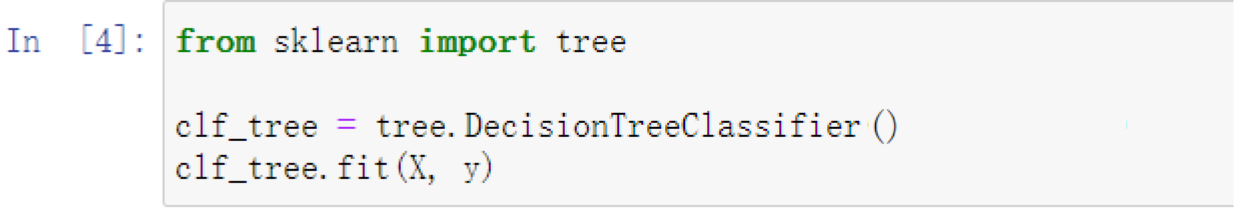
决策树分类



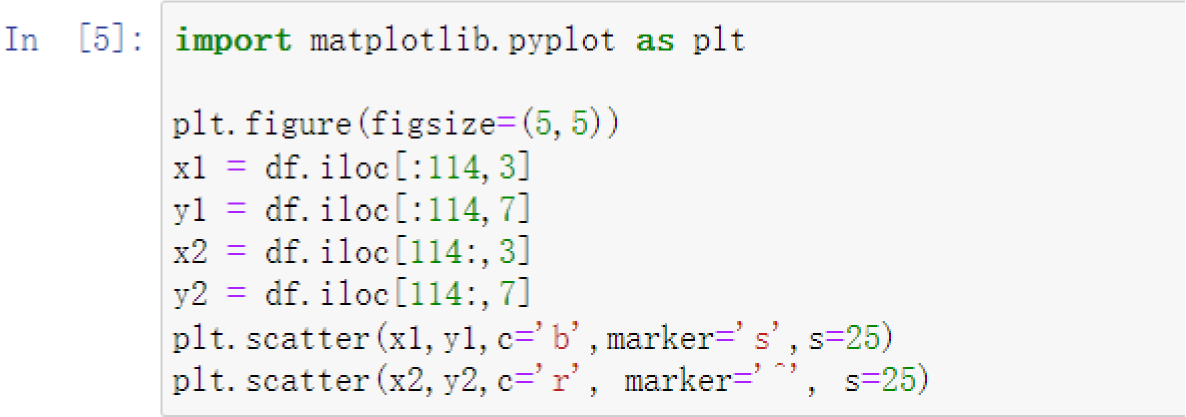
```
In [2]: df['Gap'] = df['ELUMO'] - df['EHOMO']
        df
```



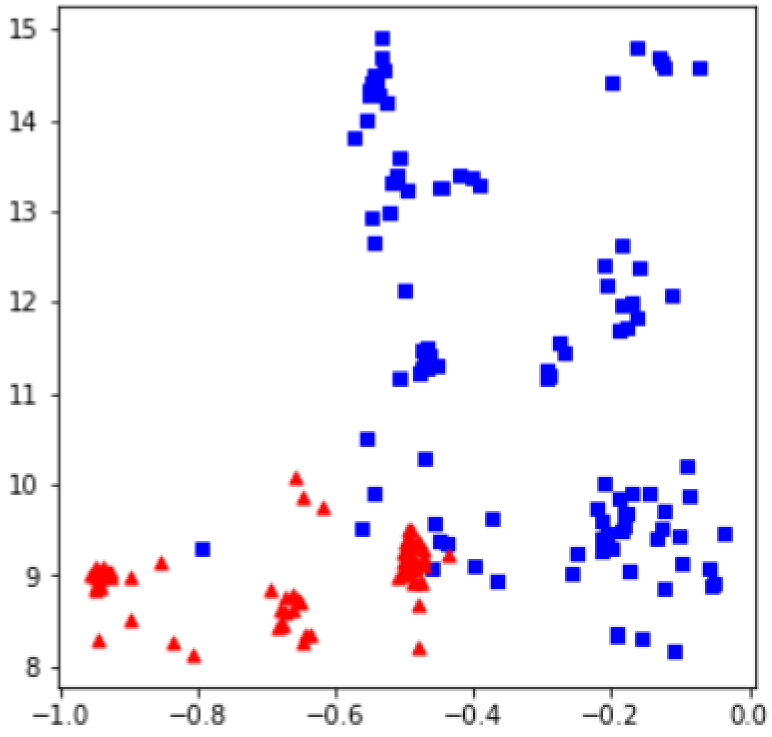
```
In [3]: X = df[[ 'Q-','Gap']]
        y = df['Class'].values
```



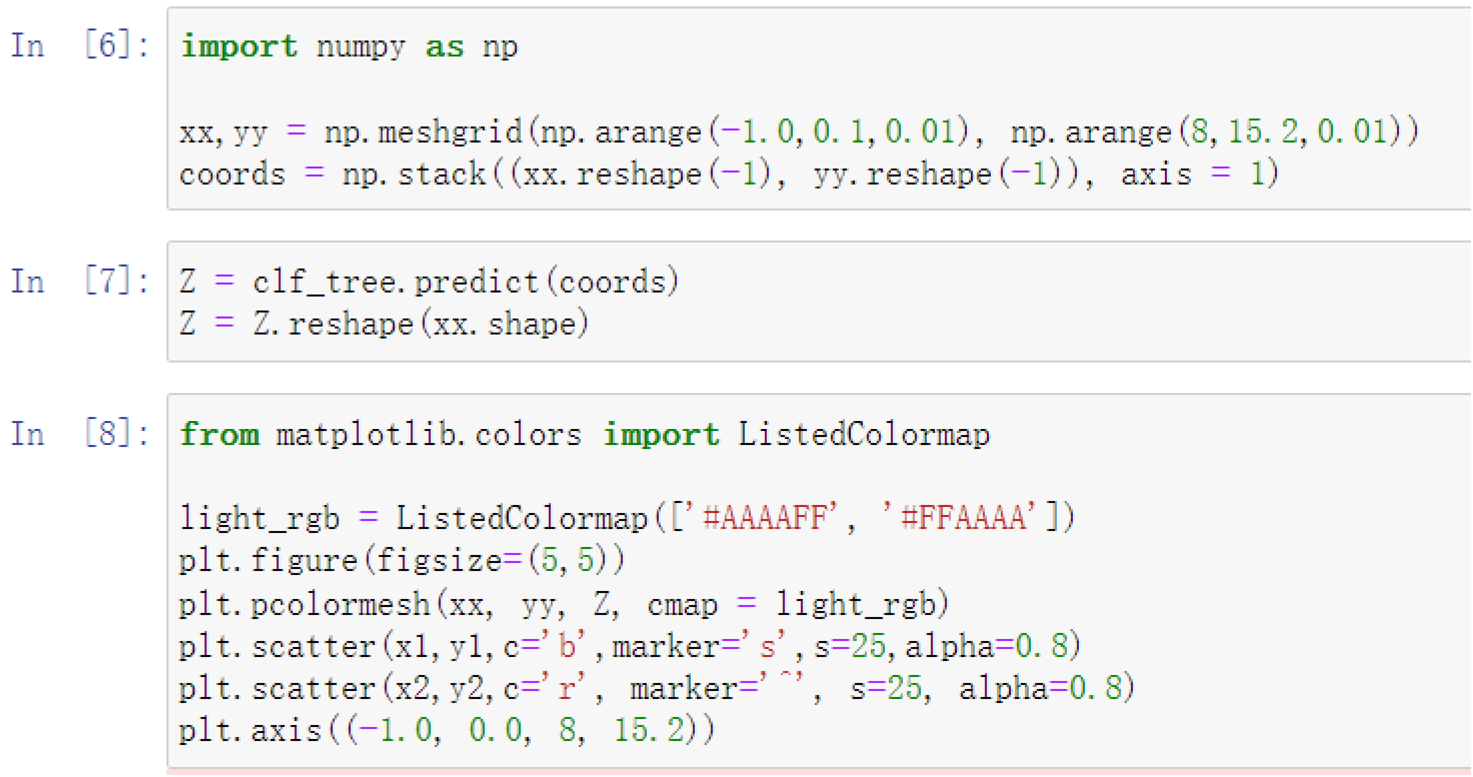
```
In [4]: from sklearn import tree

        clf_tree = tree.DecisionTreeClassifier()
        clf_tree.fit(X,  y)
```



```
In [5]: import matplotlib.pyplot as plt

        plt.figure(figsize=(5,5))
        x1 = df.iloc[:114,3]
        y1 = df.iloc[:114,7]
        x2 = df.iloc[114:,3]
        y2 = df.iloc[114:,7]
        plt.scatter(x1,y1,c='b',marker='s',s=25)
        plt.scatter(x2,y2,c='r', marker='^', s=25)
```



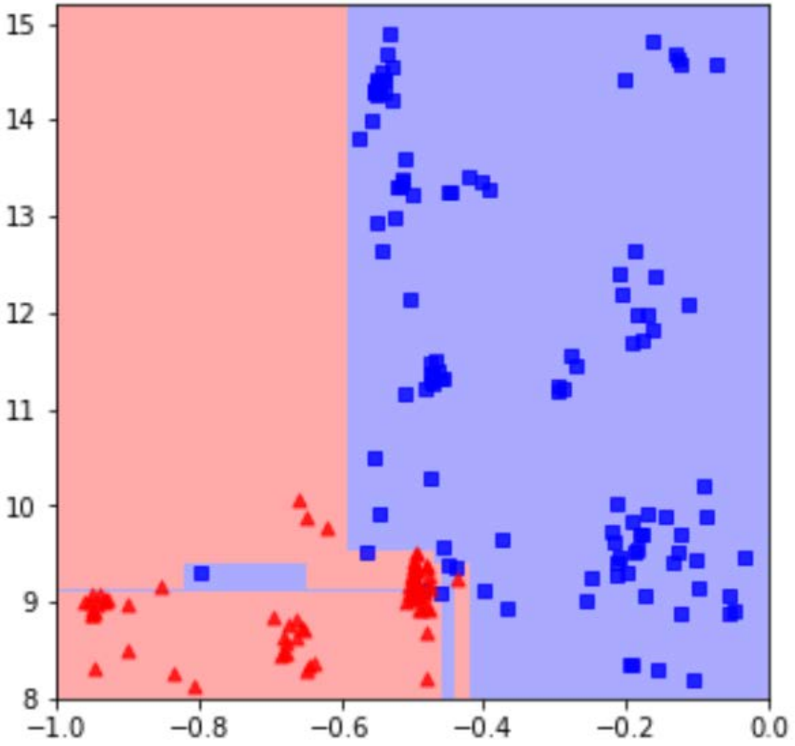
决策树分类



```
In  [6]: import numpy as np

         xx, yy = np.meshgrid(np.arange(-1.0, 0.1, 0.01), np.arange(8, 15.2, 0.01))
         coords = np.stack((xx.reshape(-1), yy.reshape(-1)), axis = 1)
```

```
In  [7]: Z = clf_tree.predict(coords)
         Z = Z.reshape(xx.shape)
```

```
In  [8]: from matplotlib.colors import ListedColormap

         light_rgb = ListedColormap(['#AAAAFF', '#FFAAAA'])
         plt.figure(figsize=(5,5))
         plt.pcolormesh(xx, yy, Z, cmap = light_rgb)
         plt.scatter(x1,y1,c='b',marker='s',s=25,alpha=0.8)
         plt.scatter(x2,y2,c='r', marker='^', s=25, alpha=0.8)
         plt.axis((-1.0, 0.0, 8, 15.2))
```

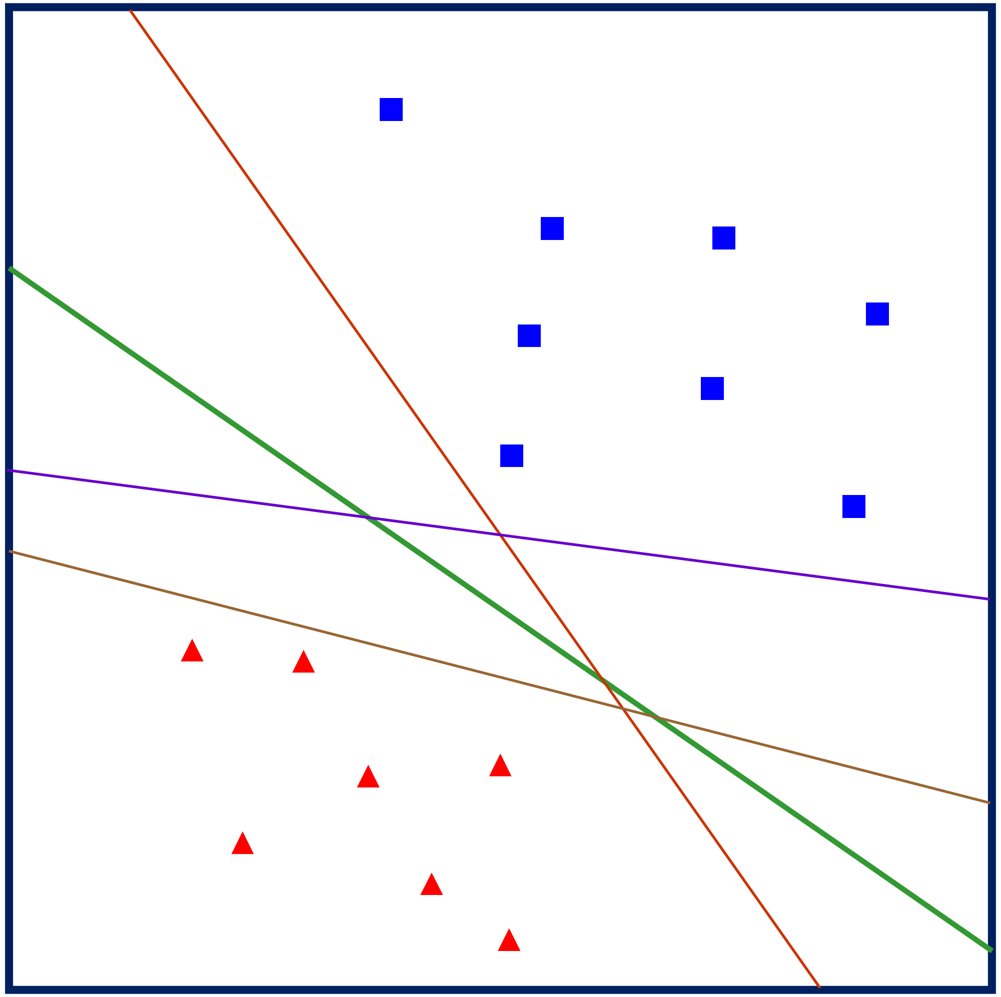


目录

1. 算法简介
2. 回归算法
3. 双金属吸附中的回归
4. 非线性回归求速率常数
5. 模型评价
6. 决策树分类算法
7. 支持向量机

支持向量机（Support Vector Machines）

问题：最佳分类方式?



1963 V. N. Vapnik and A. Y. Chervonenkis

原始支持向量机

1992 B. E. Boser, I. M. Guyon and V. N. Vapnik

核技巧

1995   C. Cortes and V. N. Vapnik

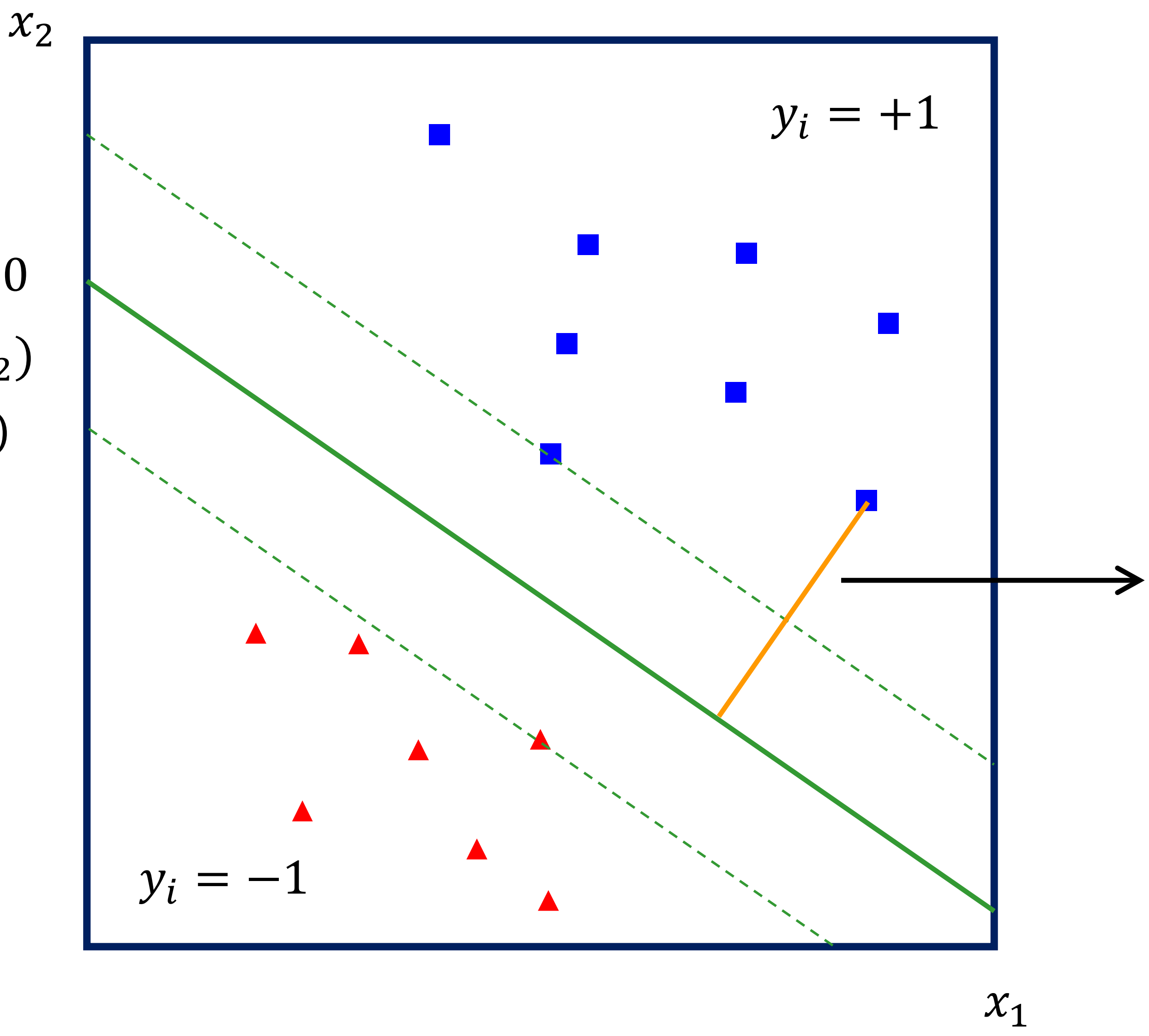
软间隔

划分超平面

划分超平面 $\boldsymbol{w}^T\boldsymbol{x} + b = 0$

$\boldsymbol{w} = (w_1, w_2)$

$\boldsymbol{x} = (x_1, x_2)$



点到超平面的距离

$$r = \frac{|\boldsymbol{w}^T\boldsymbol{x} + b|}{\|\boldsymbol{w}\|}$$

支持向量

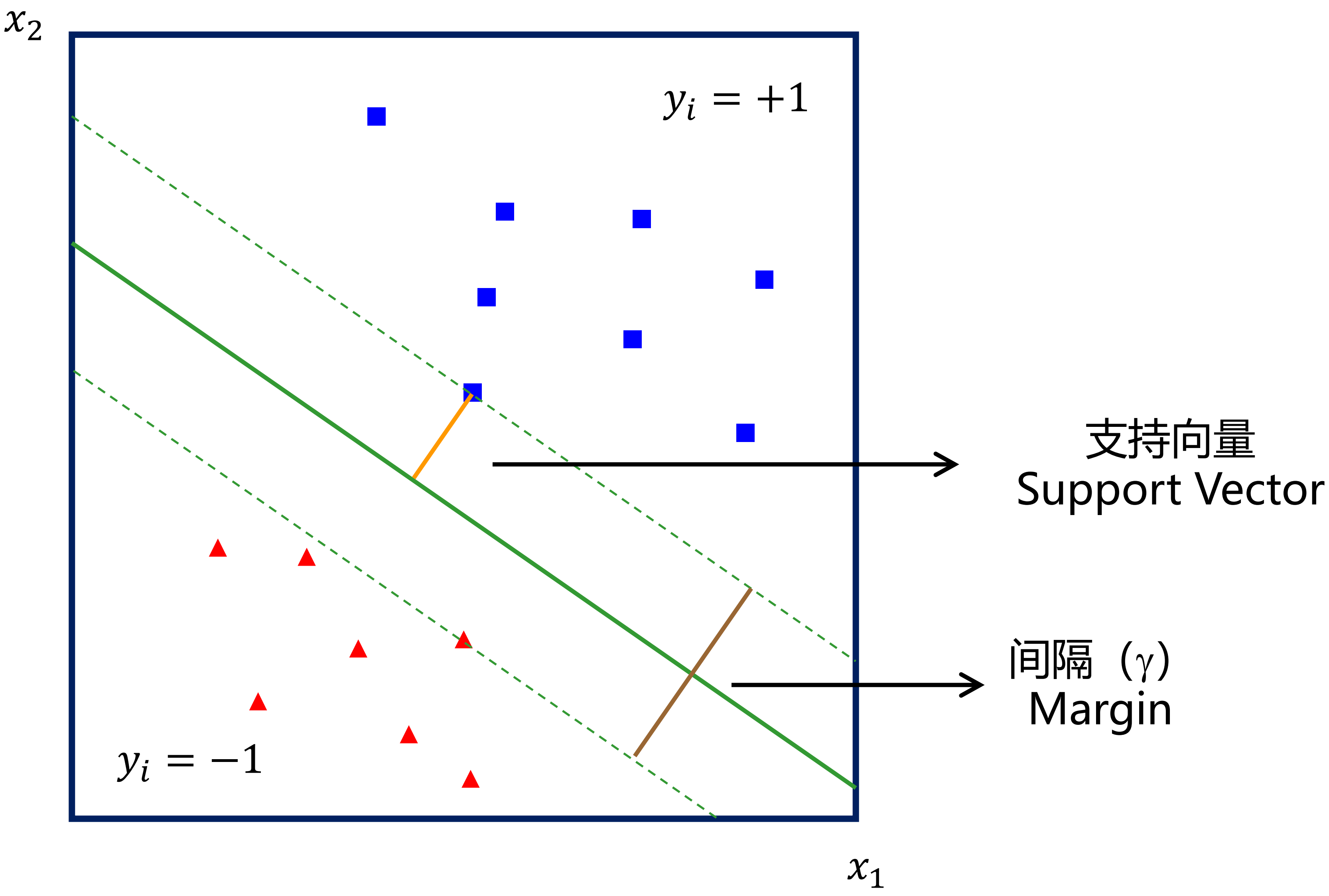
令

$$\begin{cases} \boldsymbol{w}^T\boldsymbol{x} + b \geq 1, & y_i = +1 \\ \boldsymbol{w}^T\boldsymbol{x} + b \leq -1, & y_i = -1 \end{cases}$$

等号在距离最近的点满足。

支持向量到超平面的距离之和

$$\gamma = \frac{2}{\|\boldsymbol{w}\|}$$



支持向量机（Support Vector Machines）

寻找具有最大间隔的划分超平面

$$\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|} \quad s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x}+b) \geq 1, \qquad i=1,2,3,\cdots,m.$$

支持向量机

$$\boxed{\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x}+b) \geq 1, \qquad i=1,2,3,\cdots,m.}$$

对偶问题

添加拉格朗日乘子 $\alpha_i \geq 0$

拉格朗日函数为

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum \alpha_i(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

L对*w*, *b*偏导为0

$$\boldsymbol{w} = \sum \alpha_i y_i \boldsymbol{x}_i$$

$$0 = \sum \alpha_i y_i$$

对偶问题为

$$\max_{\alpha}(\sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j) \qquad s.t. \sum \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \qquad i = 1,2,\cdots,m.$$

KKT条件

模型变为

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^T \boldsymbol{x} + b \\ &= \sum \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b \end{aligned}$$

KKT（Karush-Kuhn-Tucker）条件

$$\begin{cases} \alpha_i \geq 0; \\ y_i f(x_i) - 1 \geq 0; \\ \alpha_i (y_i f(x_i) - 1) = 0. \end{cases}$$

SMO算法

SMO（Sequential Minimal Optimization）

选取一对需更新的变量α_i和α_j

固定α_i和α_j以外的参数，求解对偶问题更新α_i和α_j

仅考虑α_i和α_j时，对偶问题的约束条件为

$$a_i y_i + a_j y_j = c, \qquad a_i \geq 0, a_j \geq 0$$

其中

$$c = -\sum_{k \neq i,j} a_k y_k$$

代入对偶问题，得到单变量二次规划问题，可以轻易解出

b的确定

对于任意支持向量(x_s, y_s)

$$y_s\left(\sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x}_S + b\right) = 1, \qquad S = \{i|\alpha_i > 0, i = 1,2,\cdots,m\}$$

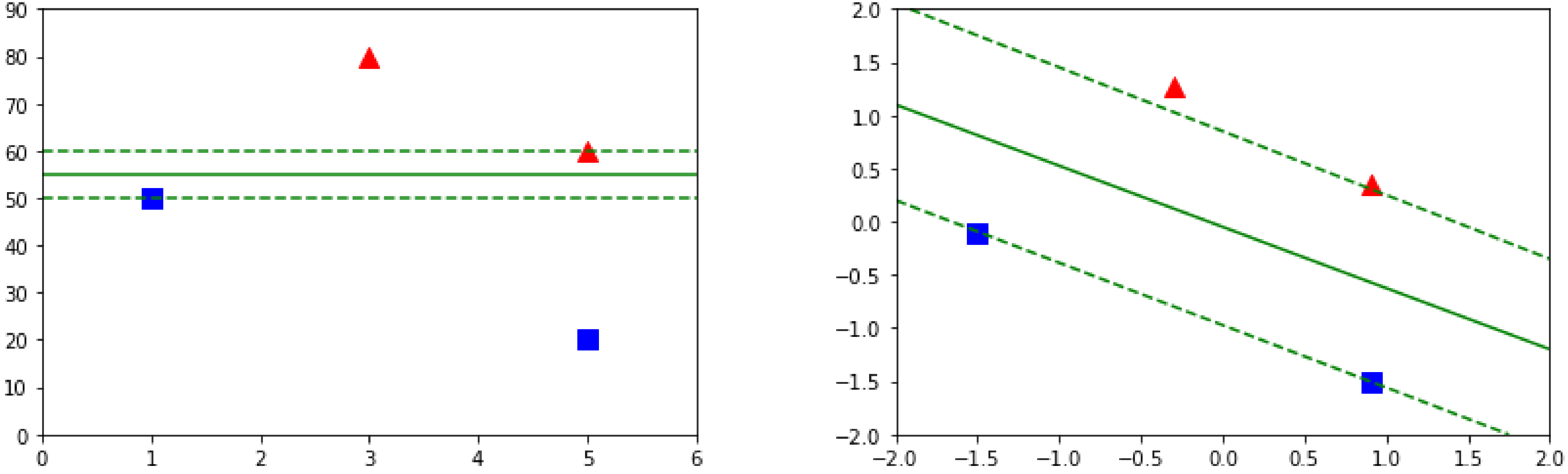
取任意支持向量(x_s, y_s)求解即可得到b

还可通过下式求解

$$b = \frac{1}{|S|}\sum_{s\in S}(y_s - \sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x}_S)$$

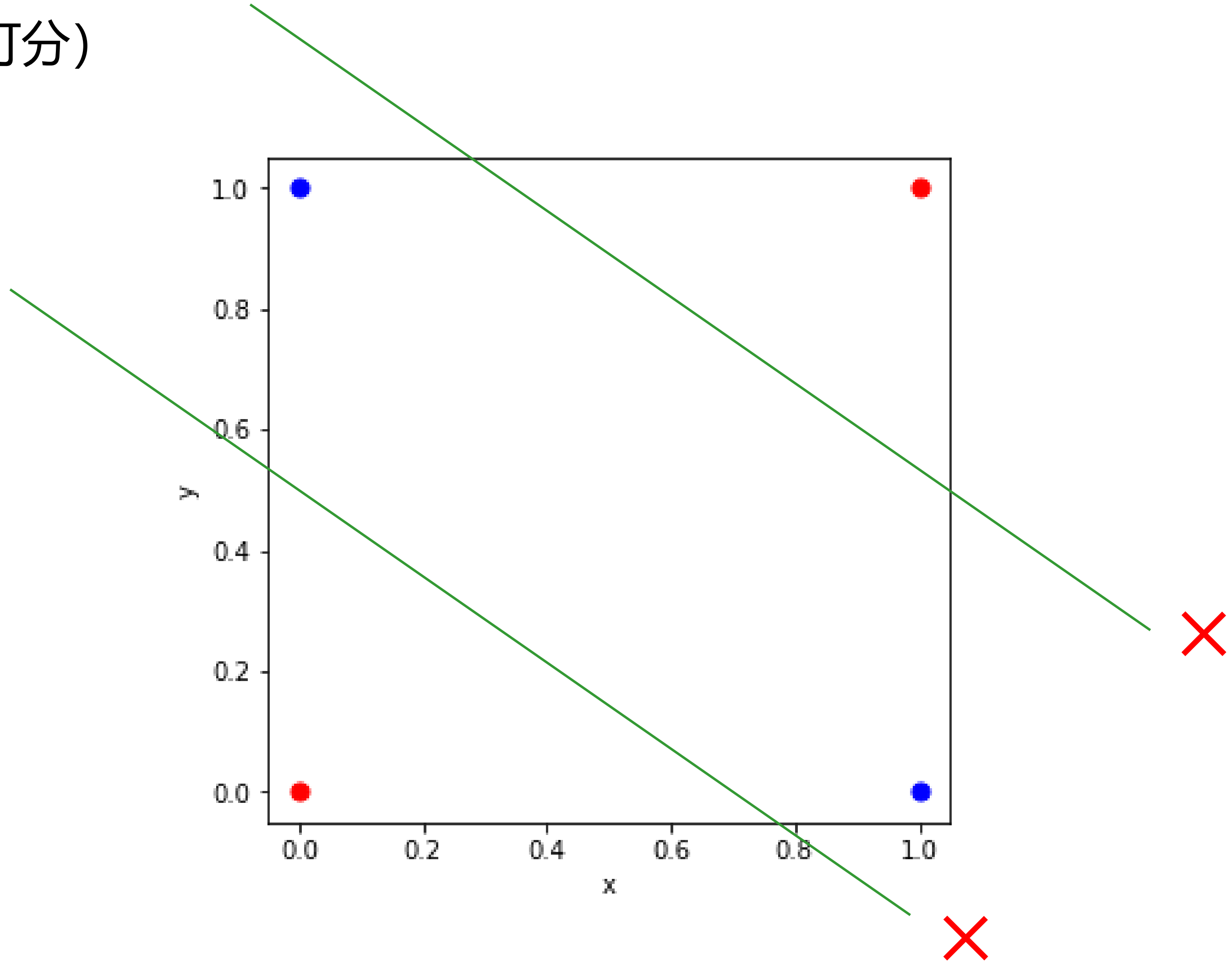
特征缩放

SVM对特征缩放敏感

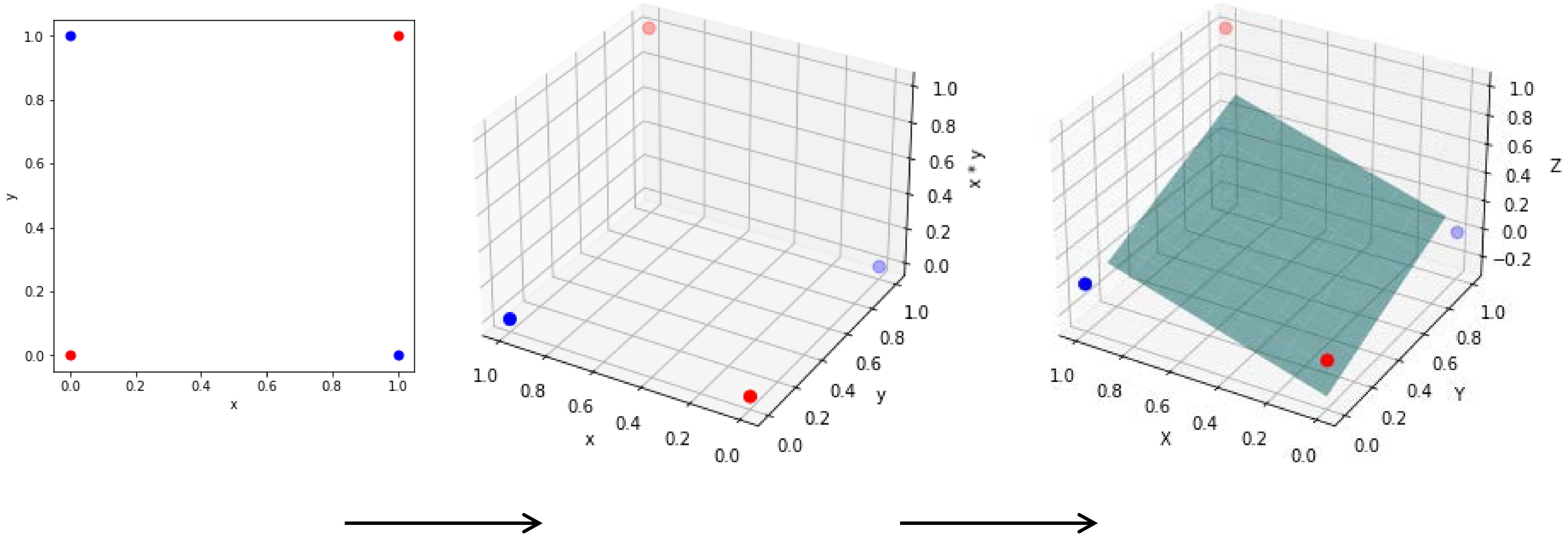


非线性问题

异或问题（线性不可分）



高维特征空间



添加新特征 x*y

SVM构建划分超平面

高维空间对偶问题

令$\phi(\boldsymbol{x})$表示映射函数，则划分超平面对应的模型可表示为

$$f(\boldsymbol{x}) = \ w^T\phi(\boldsymbol{x}) + b$$

类似的，我们需要

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 1, \qquad i = 1,2,3,\cdots,m.$$

对偶问题是

$$\max_{\alpha}\Big(\sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i\alpha_j y_i y_j \underline{\phi(\boldsymbol{x_i})^T\phi(\boldsymbol{x}_j)}\Big) \qquad s.t.\ \sum \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \qquad i = 1,2,\cdots,m.$$

高维特征空间内积

核技巧

构建核函数

$$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle=\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$$

对偶问题

$$\max_{\alpha}\sum\alpha_i-\frac{1}{2}\sum\sum\alpha_i\alpha_jy_iy_j\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)\qquad s.t.\ \sum\alpha_iy_i=0,$$

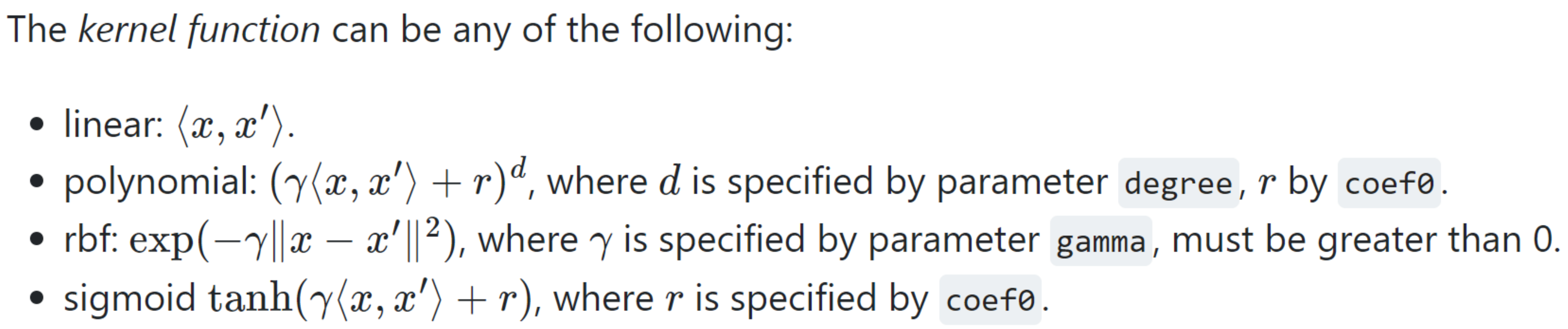
$$\alpha_i\geq 0,\qquad i=1,2,\cdots,m.$$

求解后即可得到

$$f(\boldsymbol{x})=\sum\alpha_iy_i\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)+b$$

常用核函数

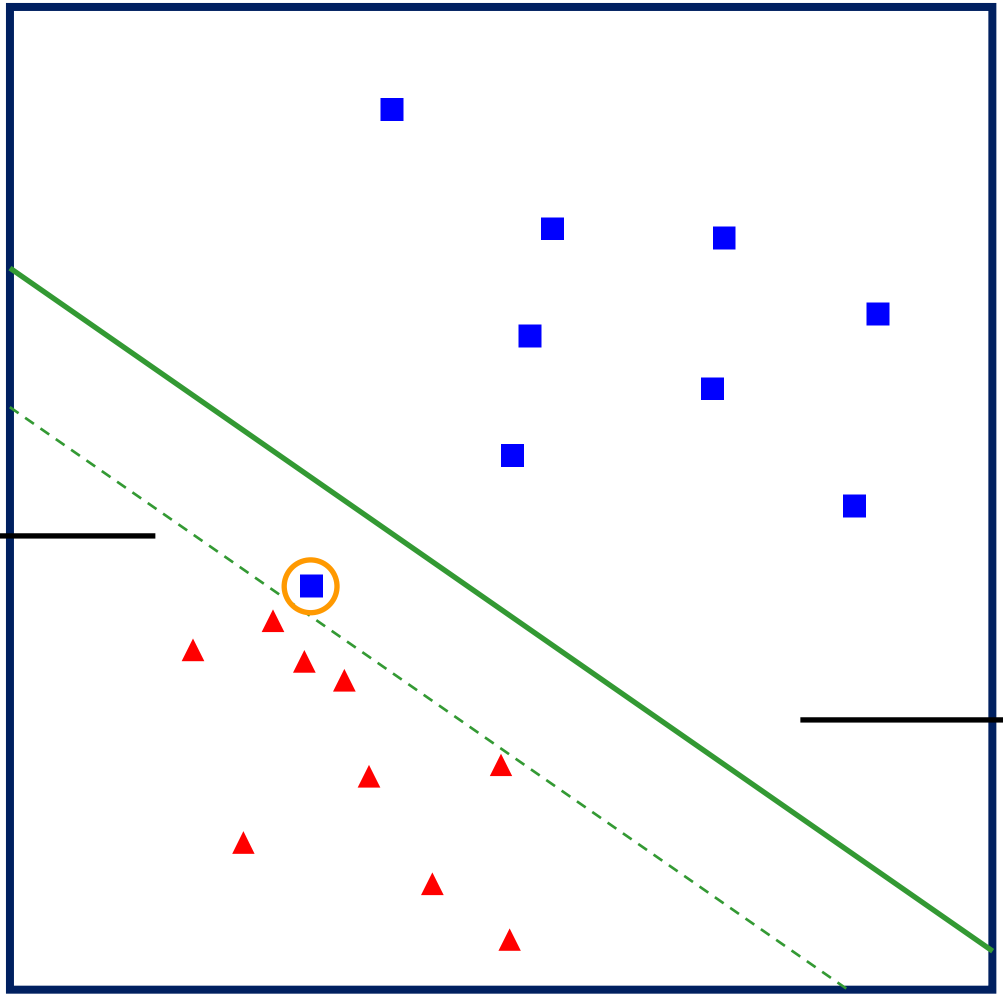
scikit-learn中的核函数



The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$, where r is specified by `coef0`.

软间隔



硬间隔

软间隔：允许部分样本不满足 $y_i(\boldsymbol{w}^T\boldsymbol{x}+b)\geq 1,$

软间隔支持向量机

优化目标改为

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum \iota_{0/1}(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

C为惩罚系数， $\iota_{0/1}$为0/1损失函数

$$\iota_{0/1}(z) = \begin{cases} 1, & if \;\; y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 0; \\ 0, & otherwise. \end{cases}$$

hinge损失函数

$$\iota_{hinge}(z) = \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

优化目标

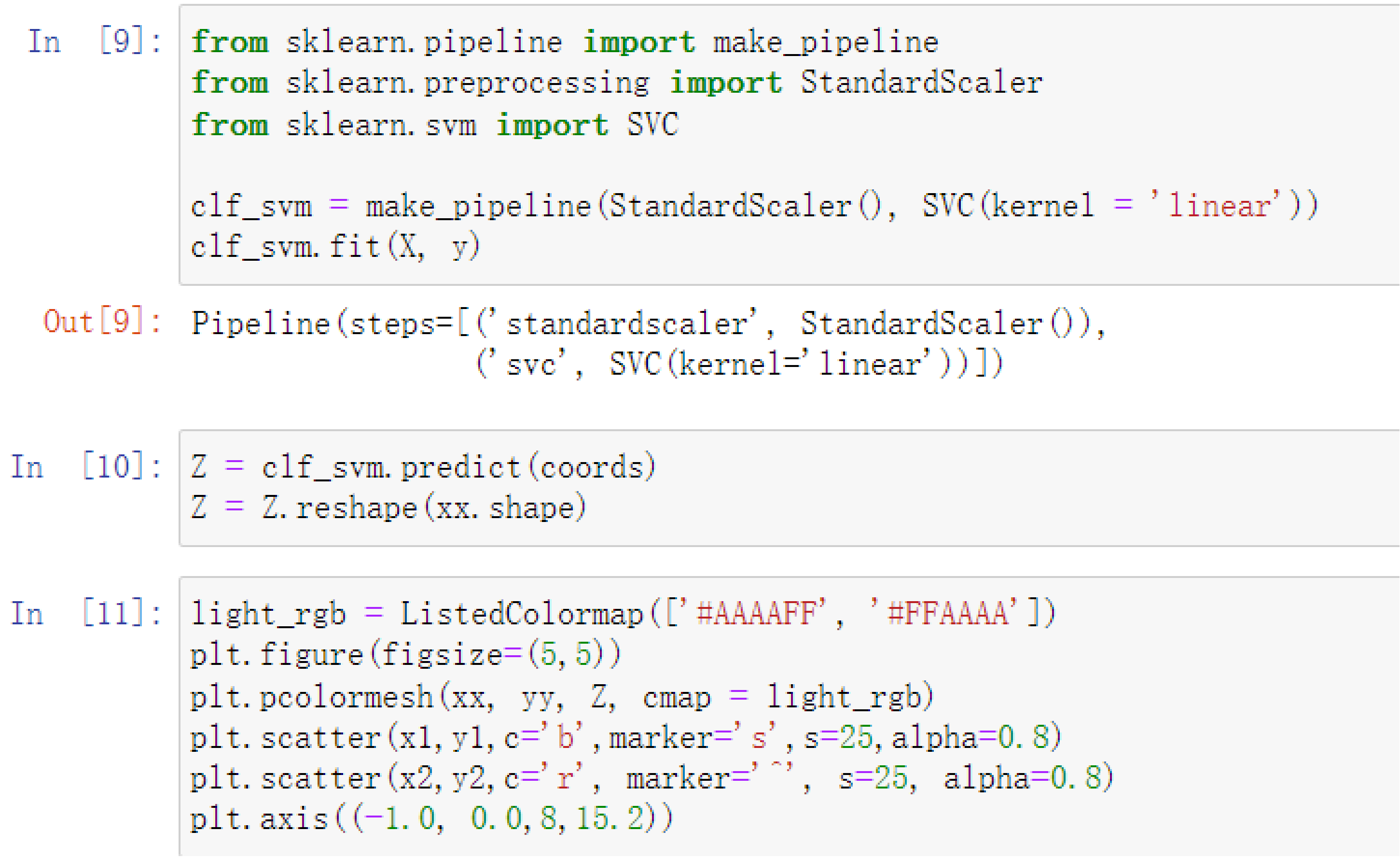
$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

松弛变量$\xi_i \geq 0$

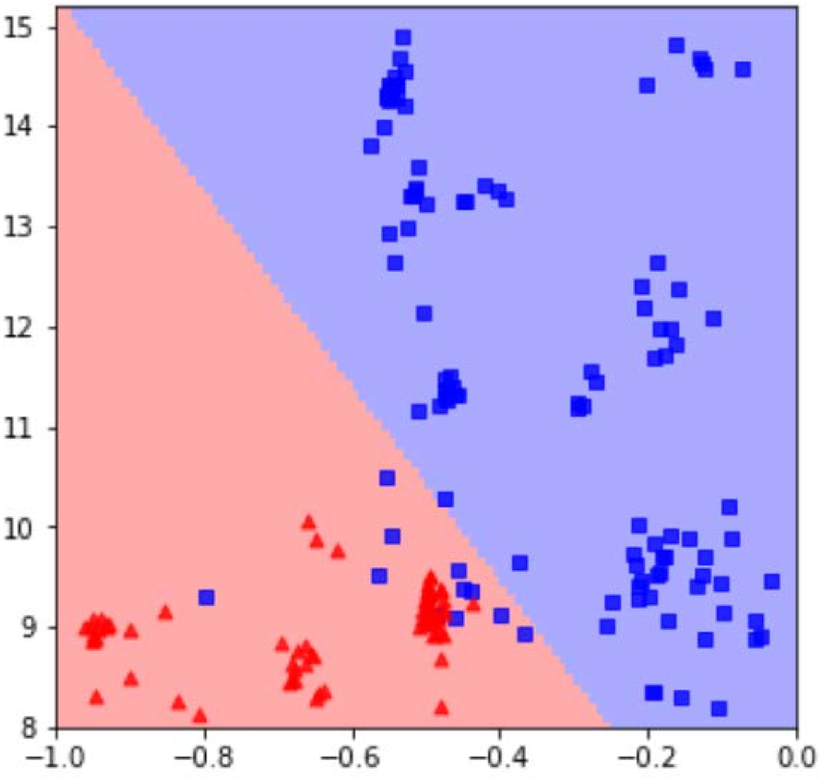
$$\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum \xi_i \qquad s.t. \;\; y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \qquad i = 1,2,\cdots,m.$$

支持向量机（Support Vector Machines）



```
In  [9]: from sklearn.pipeline import make_pipeline
         from sklearn.preprocessing import StandardScaler
         from sklearn.svm import SVC

         clf_svm = make_pipeline(StandardScaler(), SVC(kernel = 'linear'))
         clf_svm.fit(X, y)

Out[9]:  Pipeline(steps=[('standardscaler', StandardScaler()),
                         ('svc', SVC(kernel='linear'))])

In [10]: Z = clf_svm.predict(coords)
         Z = Z.reshape(xx.shape)

In [11]: light_rgb = ListedColormap(['#AAAAFF', '#FFAAAA'])
         plt.figure(figsize=(5,5))
         plt.pcolormesh(xx, yy, Z, cmap = light_rgb)
         plt.scatter(x1,y1,c='b',marker='s',s=25,alpha=0.8)
         plt.scatter(x2,y2,c='r', marker='^', s=25, alpha=0.8)
         plt.axis((-1.0, 0.0,8,15.2))
```


15
14
13
12
11
10
9
8
−1.0
−0.8
−0.6
−0.4
−0.2
0.0