

声明：本课程版权归华算科技所有，仅限个人学习，严禁任何形式的录制、传播和账号分享。一经发现，平台将依法保留追究权，情节严重者将承担法律责任。

Python与机器学习

——机器学习经典案例1

华算科技 黄老师
2022年2月24日



1. 实操：预测d带中心

利用机器学习方法预测金属与双金属的d带中心



RSC Advances

PAPER

[View Article Online](#)

[View Journal](#) | [View Issue](#)



Cite this: *RSC Adv.*, 2016, 6, 52587

Machine-learning prediction of the d-band center for metals and bimetals

Ichigaku Takigawa,^{*ab} Ken-ichi Shimizu,^{cd} Koji Tsuda^{efg} and Satoru Takakusagi^c

The d-band center for metals has been widely used in order to understand activity trends in metal-surface-catalyzed reactions in terms of the linear Brønsted–Evans–Polanyi relation and Hammer–Nørskov d-band model. In this paper, the d-band centers for eleven metals (Fe, Co, Ni, Cu, Ru, Rh, Pd, Ag, Ir, Pt, Au) and their pairwise bimetals for two different structures (1% metal doped- or overlayer-covered metal surfaces) are statistically predicted using machine learning methods from readily available values as descriptors for the target metals (such as the density and the enthalpy of fusion of each metal). The predictive accuracy of four regression methods with different numbers of descriptors and different test-set/training-set ratios are quantitatively evaluated using statistical cross validations. It is shown that the d-band centers are reasonably well predicted by the gradient boosting regression (GBR) method with only six descriptors, even when we predict 75% of the data from only 25% given for training (average root mean square error (RMSE) < 0.5 eV). This demonstrates a potential use of machine learning methods for predicting the activity trends of metal surfaces with a negligible CPU time compared to first-principles methods.

Received 18th February 2016

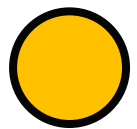
Accepted 23rd May 2016

DOI: 10.1039/c6ra04345c

www.rsc.org/advances

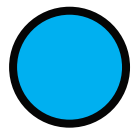
Ichigaku. T, Satoru. T, *et al.* *RSC Adv.*, 2016, 6, 52587.

输入：主客体金属描述符



原子数

价电子数



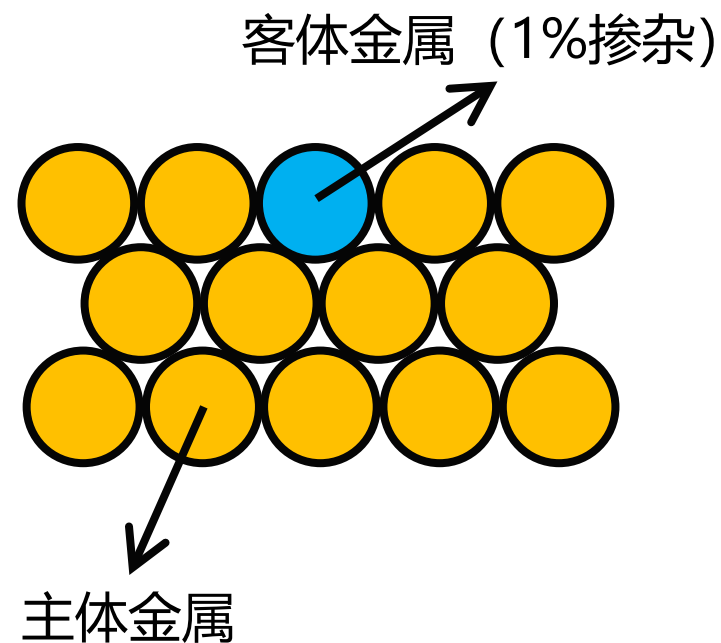
周期

密度

.....

机器学习

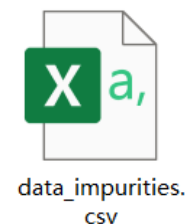
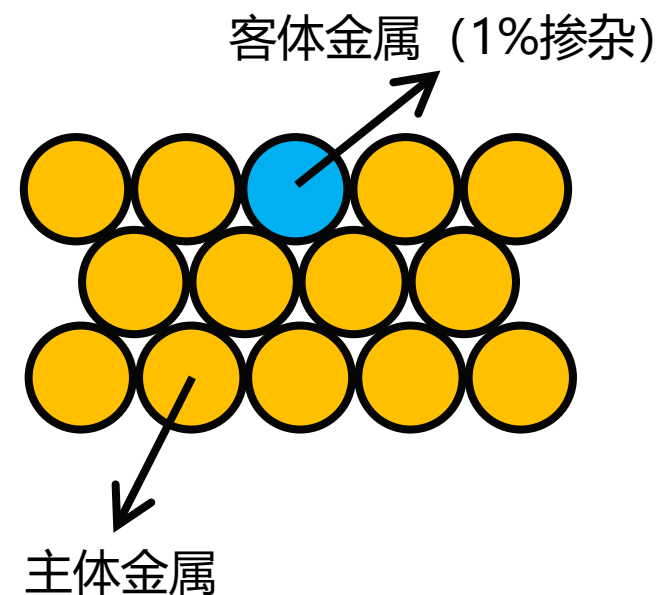
输出：双金属d带中心数值



d带中心数据集

Table 1
Shifts in d-band centers of surface impurities and overlayers relative to the clean metal values (italic)

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	<i>-0.92</i>	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
		0.14	-0.04	-0.05	-0.73	-0.72	-1.32	-1.25	-0.95	-1.48	-2.19
Co	0.01	<i>-1.17</i>	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
	-0.01		-0.20	-0.06	-0.70	-0.95	-1.65	-1.36	-1.09	-1.89	-2.39
Ni	0.09	0.19	<i>-1.29</i>	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
	0.96	0.11		0.12	-0.63	-0.74	-1.32	-1.14	-0.86	-1.53	-2.10
Cu	0.56	0.60	0.27	<i>-2.67</i>	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
	0.25	0.38	0.18		-0.22	-0.27	-1.04	-1.21	-0.32	-1.15	-1.96
Ru	0.21	0.26	0.01	0.12	<i>-1.41</i>	-0.17	-0.82	-0.27	0.02	-0.62	-0.84
	0.30	0.37	0.29	0.30		-0.12	-0.47	-0.40	-0.13	-0.61	-0.86
Rh	0.24	0.34	0.16	0.44	0.04	<i>-1.73</i>	-0.54	0.07	0.17	-0.35	-0.49
	0.31	0.41	0.34	0.22	0.03		-0.39	-0.08	0.03	-0.45	-0.57
Pd	0.37	0.54	0.50	0.94	0.24	0.36	<i>-1.83</i>	0.59	0.53	0.19	0.17
	0.36	0.54	0.54	0.80	-0.11	0.25		0.15	0.31	0.04	-0.14
Ag	0.72	0.84	0.67	0.47	0.84	0.86	0.14	<i>-4.30</i>	1.14	0.50	-0.15
	0.55	0.74	0.68	0.62	0.50	0.67	0.27		0.80	0.37	-0.21
Ir	0.21	0.27	0.05	0.21	0.09	-0.15	-0.73	-0.13	<i>-2.11</i>	-0.56	-0.74
	0.33	0.40	0.33	0.56	-0.01	-0.03	-0.42	-0.09		-0.49	-0.59
Pt	0.33	0.48	0.40	0.72	0.14	0.23	-0.17	0.44	0.38	<i>-2.25</i>	-0.05
	0.35	0.53	0.54	0.78	0.12	0.24	0.02	0.19	0.29		-0.08
Au	0.63	0.77	0.63	0.55	0.70	0.75	0.17	0.21	0.98	0.46	<i>-3.56</i>
	0.53	0.74	0.71	0.70	0.47	0.67	0.35	0.12	0.79	0.43	

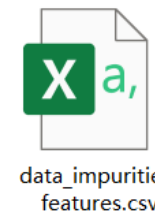


A. Ruban, J. K. Nørskov, *et al.* *J. Mol. Catal. A: Chem.*, **1997**, 115, 421.

Table 3 Input features (descriptors) used for prediction of d-band centers from ref. 34^a

Metal	G	$R/\text{\AA}$	AN	AM/g mol ⁻¹	P	EN	IE/ eV	$\Delta_{\text{fus}}H/\text{J g}^{-1}$	$\rho/\text{g cm}^{-3}$
Fe	8	2.66	26	55.85	4	1.83	7.90	247.3	7.87
Co	9	2.62	27	58.93	4	1.88	7.88	272.5	8.86
Ni	10	2.60	28	58.69	4	1.91	7.64	290.3	8.90
Cu	11	2.67	29	63.55	4	1.90	7.73	203.5	8.96
Ru	8	2.79	44	101.07	5	2.20	7.36	381.8	12.10
Rh	9	2.81	45	102.91	5	2.28	7.46	258.4	12.40
Pd	10	2.87	46	106.42	5	2.20	8.34	157.3	12.00
Ag	11	3.01	47	107.87	5	1.93	7.58	104.6	10.50
Ir	9	2.84	77	192.22	6	2.20	8.97	213.9	22.50
Pt	10	2.90	78	195.08	6	2.20	8.96	113.6	21.50
Au	11	3.00	79	196.97	6	2.40	9.23	64.6	19.30

^a Group (G), bulk Wigner-Seitz radius (R) in \AA , atomic number (AN), atomic mass (AM) in g mol^{-1} , period (P) electronegativity (EN), ionization energy (IE) in eV, enthalpy of fusion ($\Delta_{\text{fus}}H$) in J g^{-1} , density at 25 °C (ρ) in g cm^{-3} .



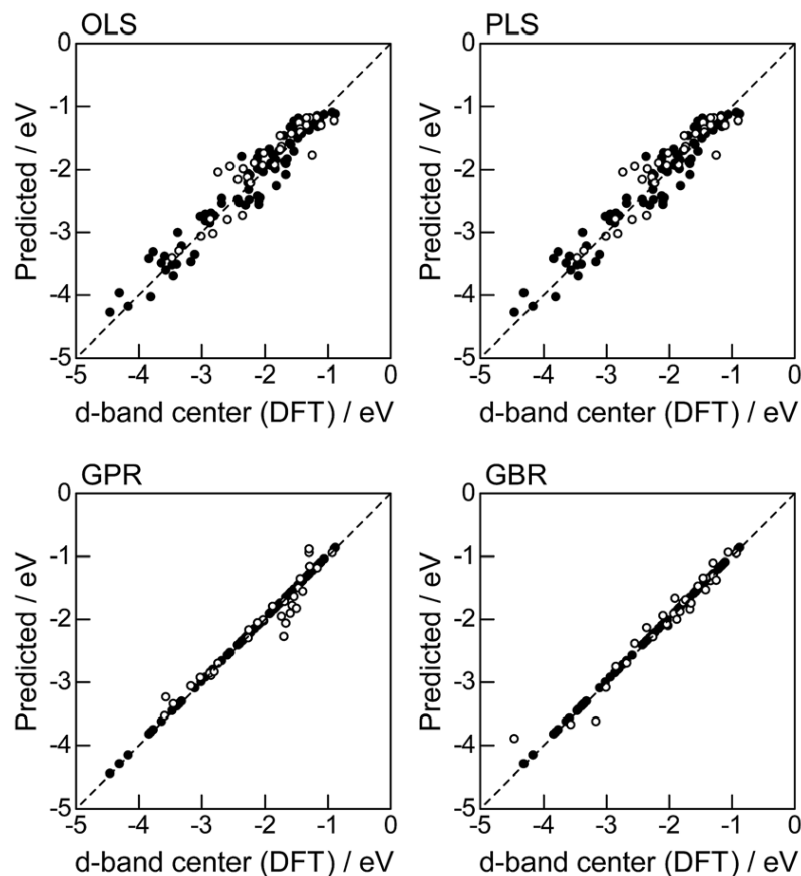


Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.

训练集：75%

测试集：25%

模型方法：

OLS：普通最小二乘回归

PLS：偏最小二乘回归

GPR：高斯过程回归

GBR：梯度提升回归

主要步骤

1. 导入待回归数据(data_impurities.csv)

1.1. 导入数据

1.2. 将d带中心数据从相对值改为绝对值

2. 导入描述符数据(data_impurities_features.csv)

3. 设置输入输出

4. 线性回归

4.1. 回归寻找规律

4.2. 可视化

4.3. 交叉验证

5. 梯度提升回归

5.1. 回归寻找规律

5.2. 可视化

5.3. 交叉验证

6. 查看数据规律 (可选)


```
In [1]: import numpy as np  
import pandas as pd
```

```
In [2]: f = open('data_impurities.csv', encoding = 'UTF-8')  
df = pd.read_csv(f, index_col = 0)  
df
```

Out[2]:

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84
Rh	0.24	0.34	0.16	0.44	0.04	-1.73	-0.54	0.07	0.17	-0.35	-0.49
Pd	0.37	0.54	0.50	0.94	0.24	0.36	-1.83	0.59	0.53	0.19	0.17
Ag	0.72	0.84	0.67	0.47	0.84	0.86	0.14	-4.30	1.14	0.50	-0.15
Ir	0.21	0.27	0.05	0.21	0.09	-0.15	-0.73	-0.13	-2.11	-0.56	-0.74
Pt	0.33	0.48	0.40	0.72	0.14	0.23	-0.17	0.44	0.38	-2.25	-0.05
Au	0.63	0.77	0.63	0.55	0.70	0.75	0.17	0.21	0.98	0.46	-3.56

```
In [3]: df.loc['Co', 'Ni']
```

```
Out[3]: -0.28
```

```
In [4]: df.loc['Co']
```

```
Out[4]: Fe    0.01  
Co   -1.17  
Ni   -0.28  
Cu   -0.16  
Ru   -0.24  
Rh   -0.58  
Pd   -1.37  
Ag   -0.91  
Ir   -0.36  
Pt   -1.19  
Au   -1.56  
Name: Co, dtype: float64
```

```
In [5]: for i in df.index:  
        for j in df.columns:  
            if(i == 'Fe' and j == 'Fe'):  
                print('host ' + i + ', guest ' + j + ', val ' + str(df.loc[i, j]))
```

```
host Fe, guest Fe, val -0.92
```

```
In [7]: for i in df.index:
        for j in df.columns:
            if i != j:
                df.loc[i, j] += df.loc[i, i]

df
```

Out[7]:

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	-0.87	-1.12	-1.05	-1.21	-1.46	-2.16	-1.75	-1.28	-2.01	-2.34
Co	-1.16	-1.17	-1.45	-1.33	-1.41	-1.75	-2.54	-2.08	-1.53	-2.36	-2.73
Ni	-1.20	-1.10	-1.29	-1.10	-1.43	-1.60	-2.26	-1.82	-1.43	-2.09	-2.42
Cu	-2.11	-2.07	-2.40	-2.67	-2.09	-2.35	-3.31	-3.37	-2.09	-3.00	-3.76
Ru	-1.20	-1.15	-1.40	-1.29	-1.41	-1.58	-2.23	-1.68	-1.39	-2.03	-2.25
Rh	-1.49	-1.39	-1.57	-1.29	-1.69	-1.73	-2.27	-1.66	-1.56	-2.08	-2.22
Pd	-1.46	-1.29	-1.33	-0.89	-1.59	-1.47	-1.83	-1.24	-1.30	-1.64	-1.66
Ag	-3.58	-3.46	-3.63	-3.83	-3.46	-3.44	-4.16	-4.30	-3.16	-3.80	-4.45
Ir	-1.90	-1.84	-2.06	-1.90	-2.02	-2.26	-2.84	-2.24	-2.11	-2.67	-2.85
Pt	-1.92	-1.77	-1.85	-1.53	-2.11	-2.02	-2.42	-1.81	-1.87	-2.25	-2.30
Au	-2.93	-2.79	-2.93	-3.01	-2.86	-2.81	-3.39	-3.35	-2.58	-3.10	-3.56

添加描述符

```
In [9]: f = open('data_impurities_features.csv', encoding = 'UTF-8')  
feat = pd.read_csv(f, index_col = 0)  
feat
```

Out[9]:

	name	group	bulk wigner-seitz radius	atomic number	atomic mass	period	electronegativity	ionization energy(eV)	H_fus	density
Fe	Iron	8	2.66	26	55.84500	4	1.83	7.9024	247.3	7.87
Co	Cobalt	9	2.62	27	58.93320	4	1.88	7.8810	272.5	8.86
Ni	Nickel	10	2.60	28	58.69340	4	1.91	7.6398	290.3	8.90
Cu	Copper	11	2.67	29	63.54600	4	1.90	7.7264	203.5	8.96
Ru	Ruthenium	8	2.79	44	101.07000	5	2.20	7.3605	381.8	12.10
Rh	Rhodium	9	2.81	45	102.90550	5	2.28	7.4589	258.4	12.40
Pd	Palladium	10	2.87	46	106.42000	5	2.20	8.3369	157.3	12.00
Ag	Silver	11	3.01	47	107.86820	5	1.93	7.5762	104.6	10.50
Ir	Iridium	9	2.84	77	192.21700	6	2.20	8.9670	213.9	22.50
Pt	Platinum	10	2.90	78	195.07800	6	2.20	8.9588	113.6	21.50
Au	Gold	11	3.00	79	196.96655	6	2.40	9.2255	64.6	19.30

查看描述符

```
In [10]: feat.head(4)
```

```
Out[10]:
```

	name	group	bulk wigner-seitz radius	atomic number	atomic mass	period	electronegativity	ionization energy(eV)	H_fus	density
Fe	Iron	8	2.66	26	55.8450	4	1.83	7.9024	247.3	7.87
Co	Cobalt	9	2.62	27	58.9332	4	1.88	7.8810	272.5	8.86
Ni	Nickel	10	2.60	28	58.6934	4	1.91	7.6398	290.3	8.90
Cu	Copper	11	2.67	29	63.5460	4	1.90	7.7264	203.5	8.96

```
In [11]: feat.loc['Co']
```

```
Out[11]: name          Cobalt
group              9
bulk wigner-seitz radius    2.62
atomic number           27
atomic mass             58.9332
period                  4
electronegativity        1.88
ionization energy(eV)     7.881
H_fus                 272.5
density                8.86
Name: Co, dtype: object
```

设置输入输出

```
In [14]: x = list()
y = list()
for i in df.index:
    for j in df.columns:
        vec_i = feat.loc[i].to_numpy()
        vec_j = feat.loc[j].to_numpy()
        x_val = np.concatenate((vec_i, vec_j))
        y_val = df.loc[i][j]
        x.append(x_val)
        y.append(y_val)
    if i == 'Fe' and j == 'Co':
        print('host ' + i + ', guest ' + j + ', input = ' + str(x_val) + ', output = ' + str(y_val))
```

```
host Fe, guest Co, input = [ 8.          2.66    26.          55.845    4.          1.83    7.9024  247.3
 7.87    9.          2.62    27.          58.9332    4.          1.88    7.881
272.5    8.86 ], output = -0.87
```

```
In [18]: from sklearn.utils import shuffle
X_r, y_r = shuffle(X, y)
X_train, y_train = X_r[:-30, :], y_r[:-30]
X_test, y_test = X_r[-30:, :], y_r[-30:]
```

shuffle()

打乱函数

```
In [19]: from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_train_lr = lr.predict(X_train)
y_pred_test_lr = lr.predict(X_test)
```

线性回归

```
In [21]: import matplotlib.pyplot as plt
```

```
plt.figure(figsize = (5, 5))  
plt.scatter(y_train, y_pred_train_lr, alpha=0.5, color = 'blue', label = 'training')  
plt.scatter(y_test, y_pred_test_lr, alpha=0.5, color = 'red', label = 'test')  
plt.legend()  
plt.xlabel('DFT')  
plt.ylabel('prediction')
```

```
Out[21]: Text(0, 0.5, 'prediction')
```

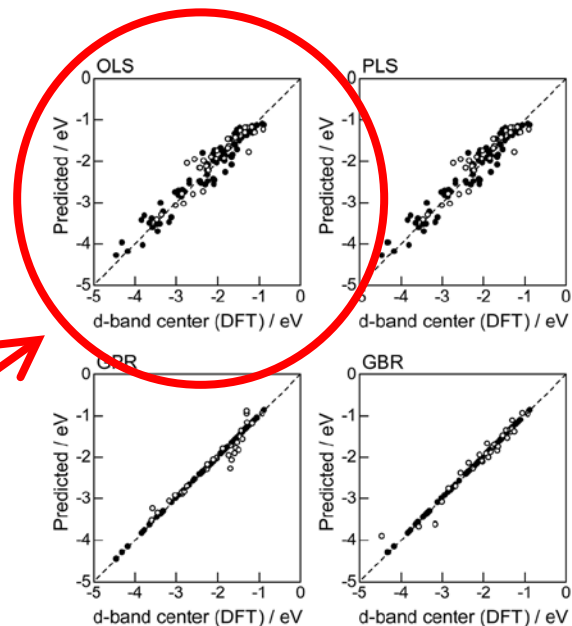
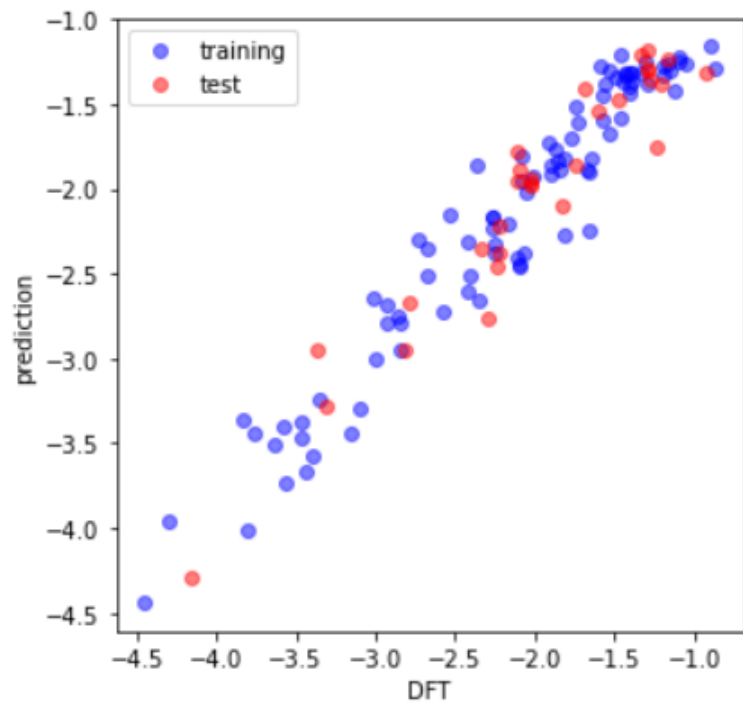


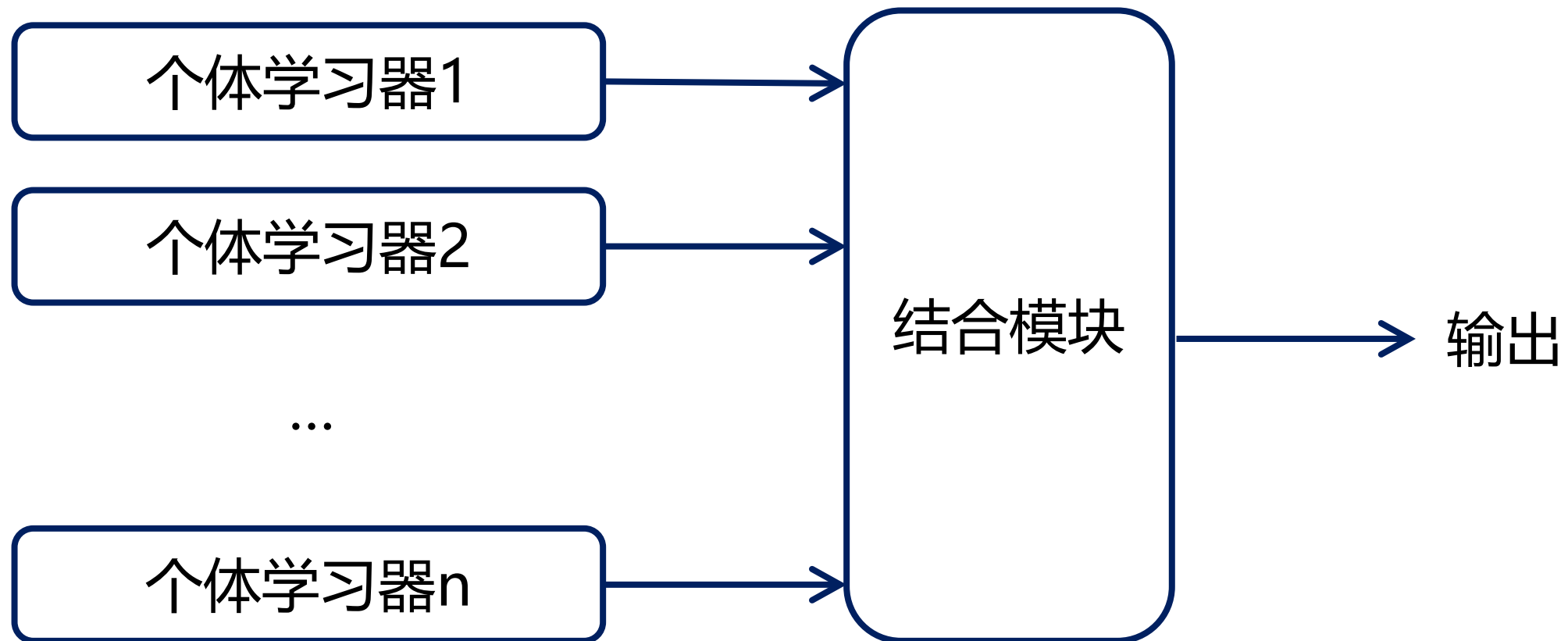
Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.


```
In [22]: from sklearn.metrics import mean_squared_error
rmse_tr_lr = mean_squared_error(y_train, y_pred_train_lr, squared=False)
rmse_te_lr = mean_squared_error(y_test, y_pred_test_lr, squared=False)
print('RMSE(training)%.3f' % rmse_tr_lr)
print('RMSE(test)%.3f' % rmse_te_lr)
```

```
RMSE(training)0.214
RMSE(test)0.213
```

```
In [23]: from sklearn.model_selection import cross_val_score, ShuffleSplit
cvf = ShuffleSplit(n_splits=100, test_size=0.25)
r2_lr = cross_val_score(lr, X, y, cv=cvf)
scores_lr = cross_val_score(lr, X, y, cv=cvf, scoring = 'neg_root_mean_squared_error')
print('100 times mean r2: %.3f' % r2_lr.mean())
print('100 times mean RMSE: %.3f' % -scores_lr.mean())
```

```
100 times mean r2: 0.897
100 times mean RMSE: 0.250
```



考虑二分类任务

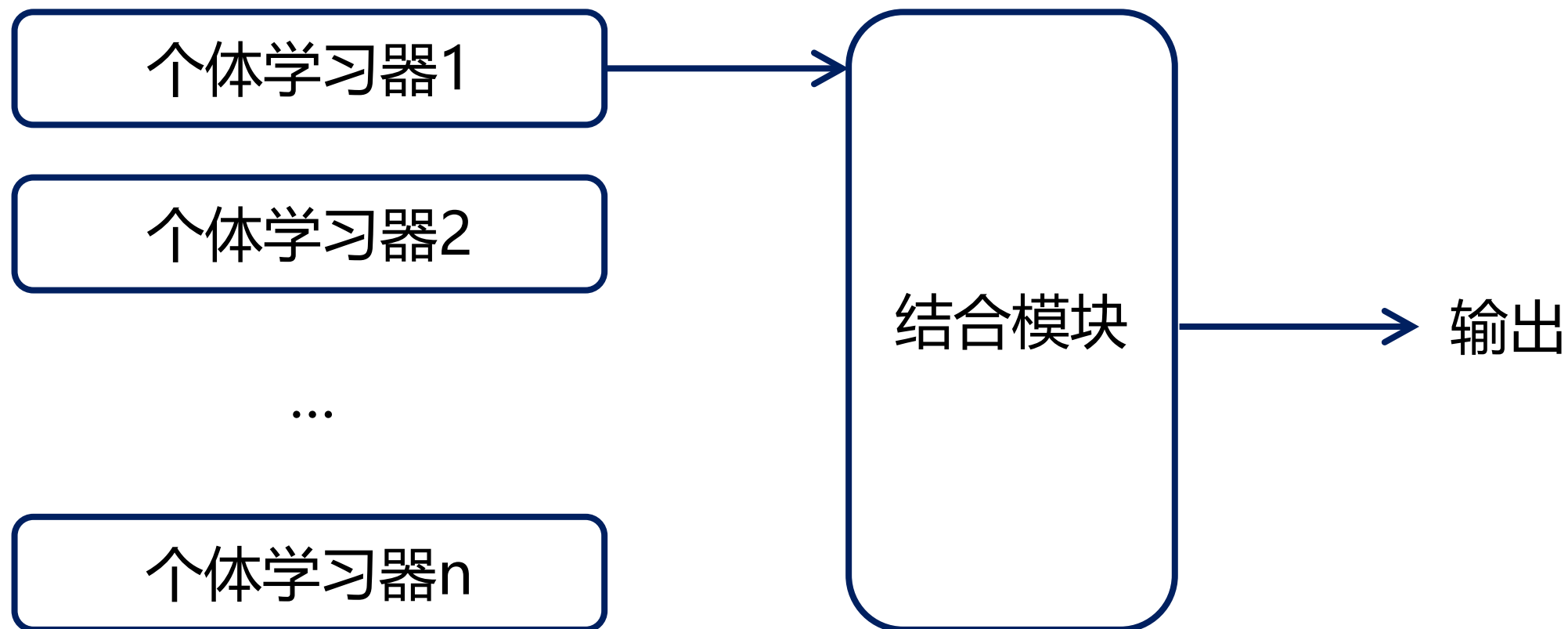
	测试例1	测试例2	测试例3
h1	√	√	×
h2	√	×	√
h3	×	√	√
集成	√	√	√

	测试例1	测试例2	测试例3
h1	√	√	×
h2	√	√	×
h3	√	√	×
集成	√	√	×

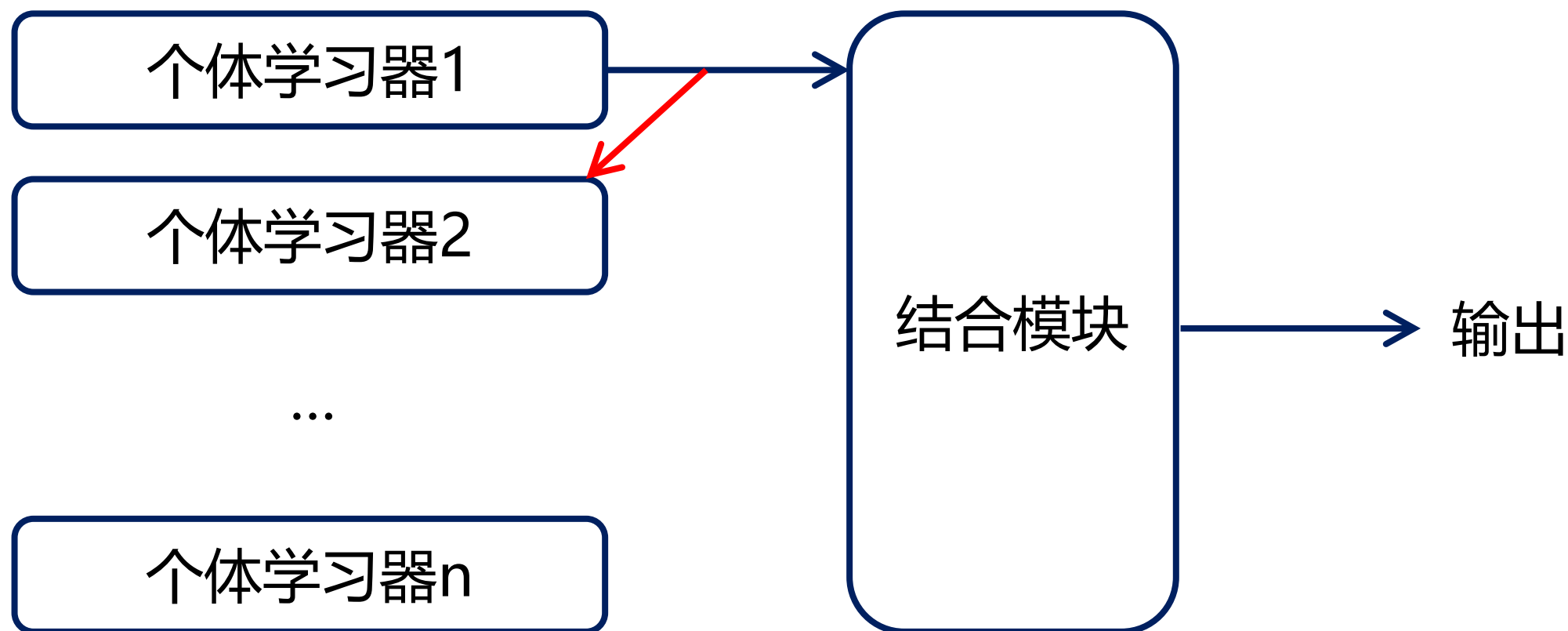
	测试例1	测试例2	测试例3
h1	√	×	×
h2	×	√	×
h3	×	×	√
集成	×	×	×

集成个体应 “好而不同”

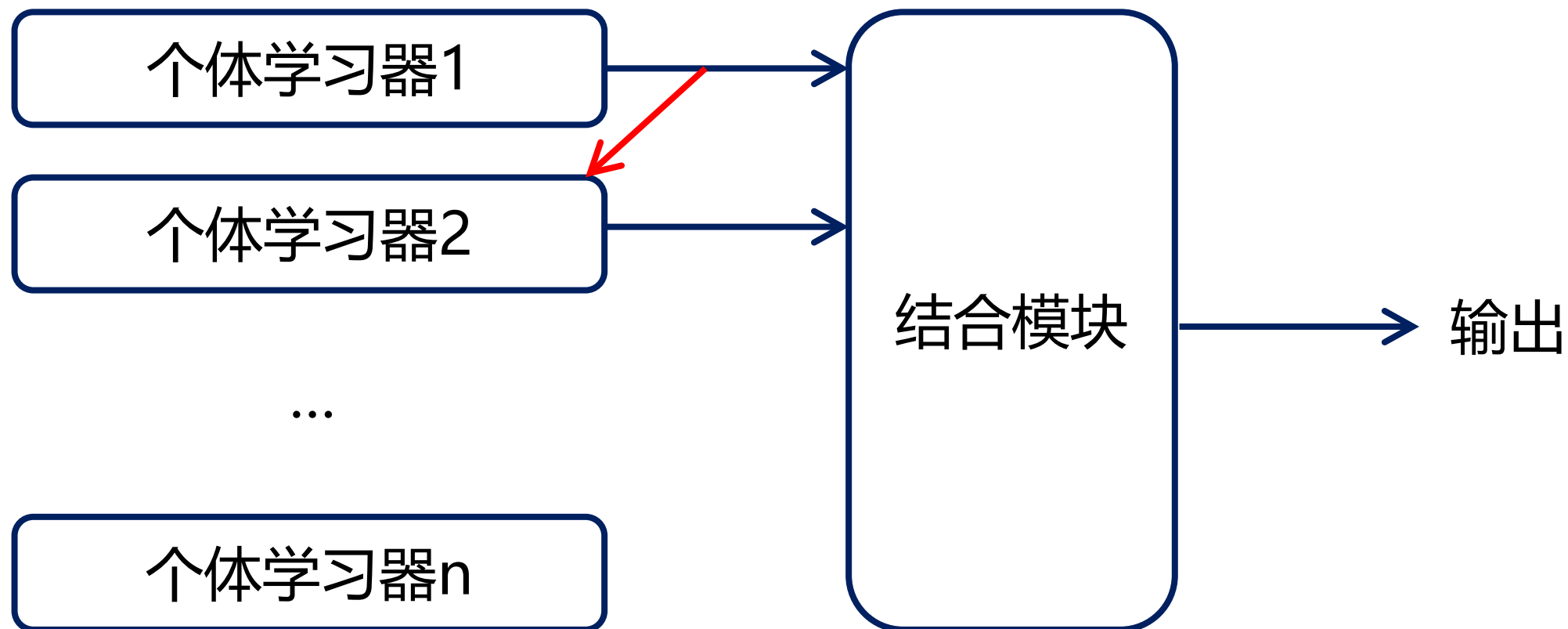
① 训练学习器1



② 调整样本权重

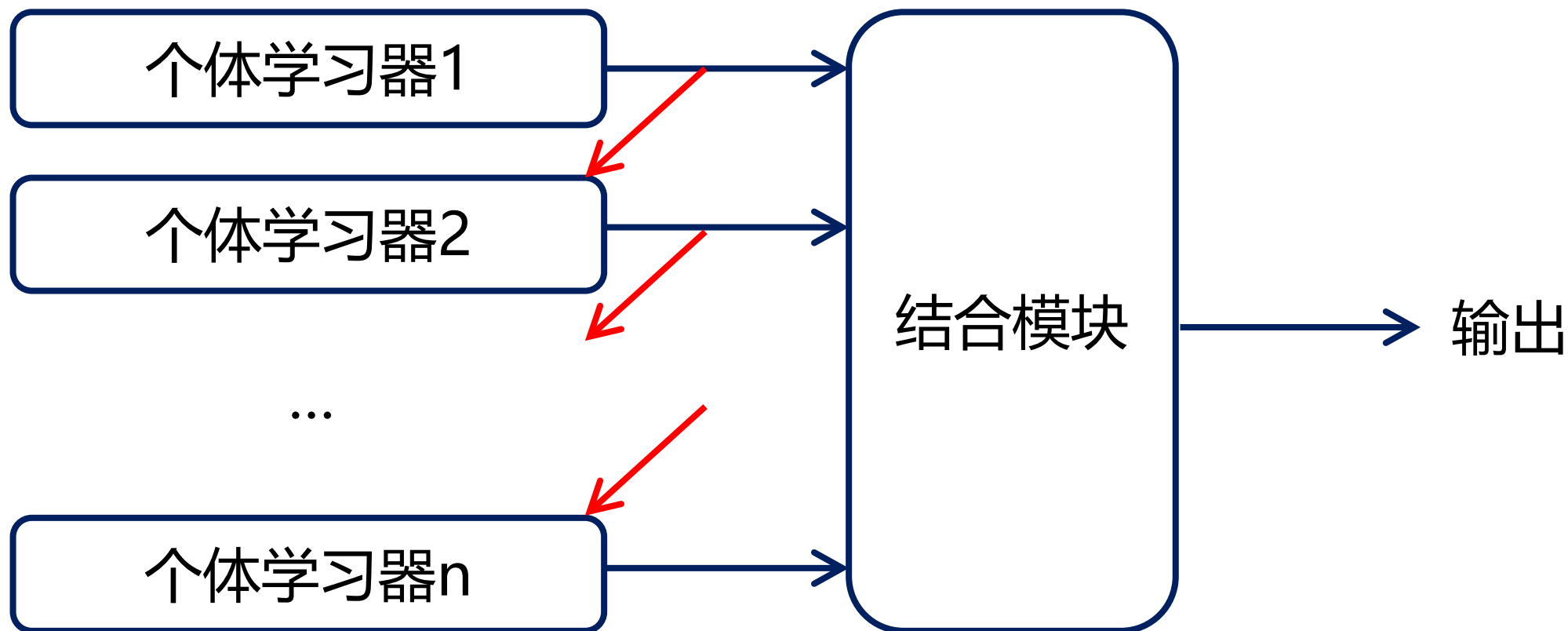


③ 训练学习器2



Boosting

得到完整模型



梯度提升回归

```
In [29]: from sklearn.ensemble import GradientBoostingRegressor  
gbr = GradientBoostingRegressor()  
gbr.fit(X_train, y_train)  
y_pred_train_gbr = gbr.predict(X_train)  
y_pred_test_gbr = gbr.predict(X_test)
```

```
In [30]: plt.figure(figsize = (5,5))  
plt.scatter(y_train, y_pred_train_gbr, alpha=0.5, color = 'blue', label = 'training')  
plt.scatter(y_test, y_pred_test_gbr, alpha=0.5, color = 'red', label = 'test')  
plt.legend()  
plt.xlabel('DFT')  
plt.ylabel('prediction')
```

Out[30]: Text(0, 0.5, 'prediction')

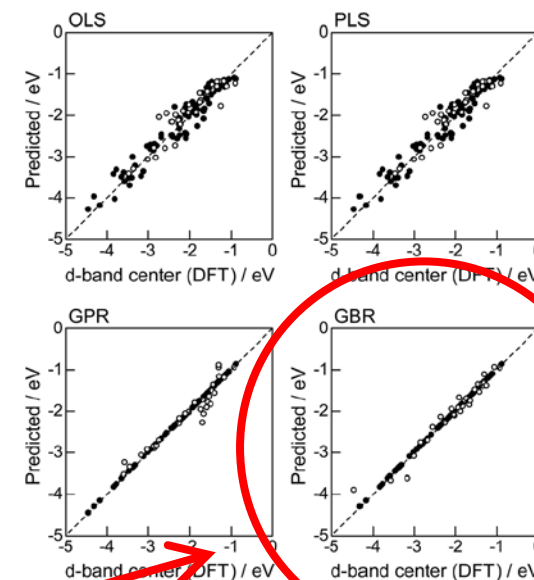
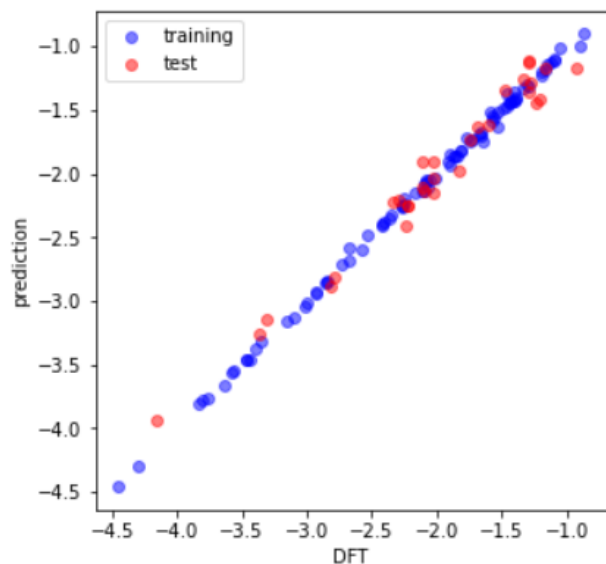


Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.

Ichigaku. T, Satoru. T, *et al.* *RSC Adv.*, **2016**, 6, 52587.


```
In [31]: rmse_tr_gbr = mean_squared_error(y_train, y_pred_train_gbr, squared=False)
rmse_te_gbr = mean_squared_error(y_test, y_pred_test_gbr, squared=False)
print('RMSE(training)%.3f' % rmse_tr_gbr)
print('RMSE(test)%.3f' % rmse_te_gbr)
```

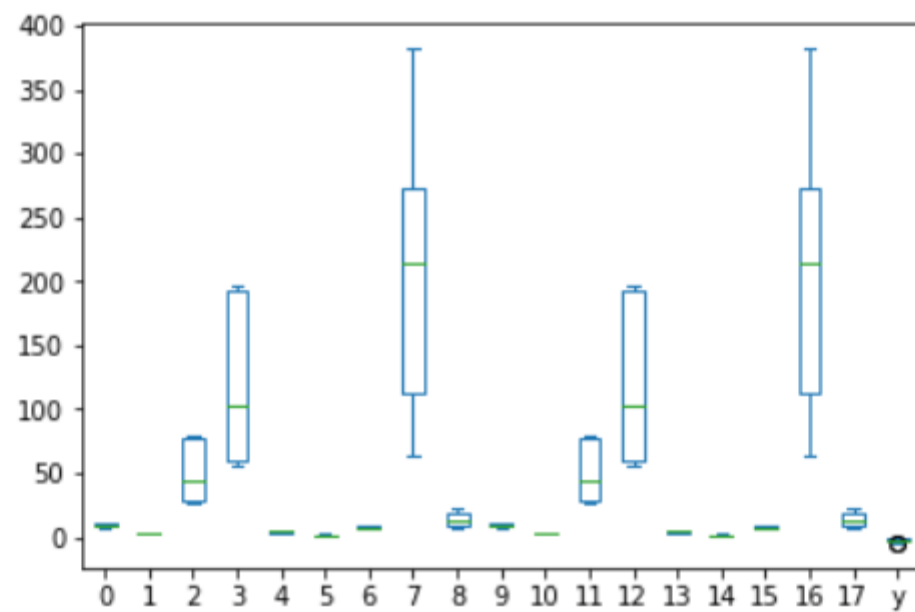
```
RMSE(training)0.032
RMSE(test)0.126
```

```
In [32]: r2_gbr = cross_val_score(gbr, X, y, cv=cvf, scoring = 'r2')
scores_gbr = cross_val_score(gbr, X, y, cv=cvf, scoring = 'neg_root_mean_squared_error')
print('100 times mean r2: %.3f' % r2_gbr.mean())
print('100 times mean RMSE: %.3f' % -scores_gbr.mean())
```

```
100 times mean r2: 0.965
100 times mean RMSE: 0.152
```

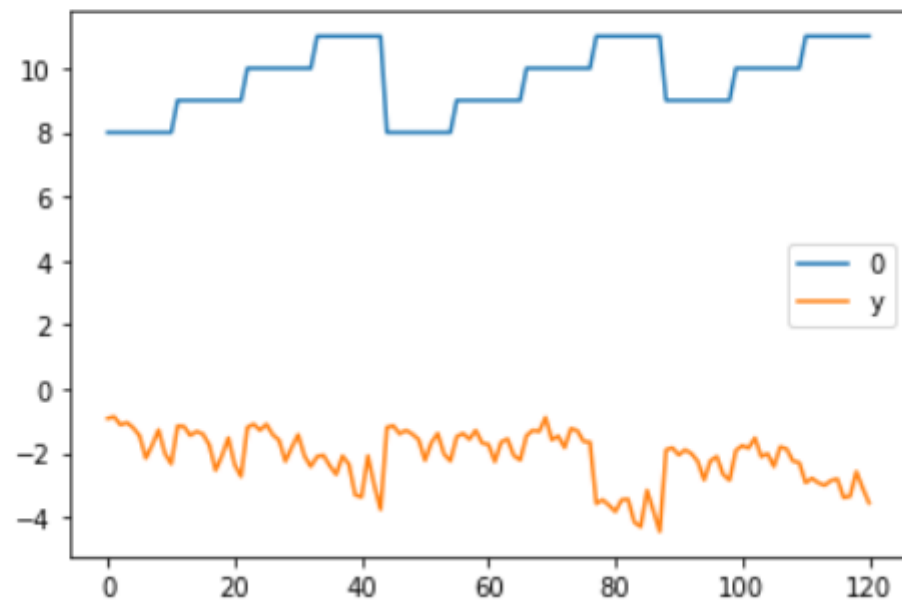
```
In [34]: df.plot.box()
```

```
Out[34]: <AxesSubplot:>
```



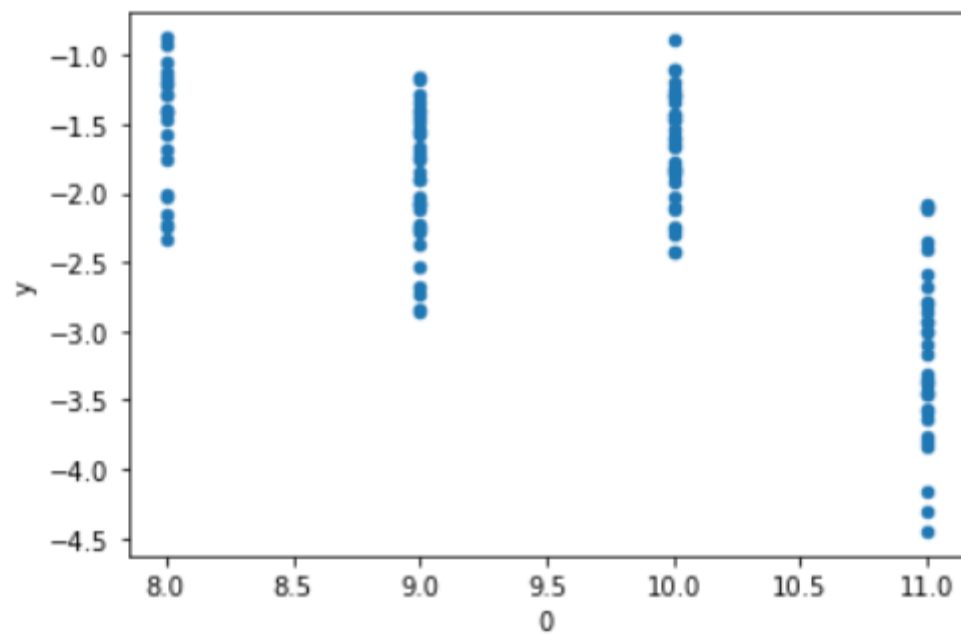
```
In [35]: idx = [0, 'y']  
df[idx].plot()
```

Out[35]: <AxesSubplot:>



```
In [36]: df.plot.scatter(x = 0, y = 'y')
```

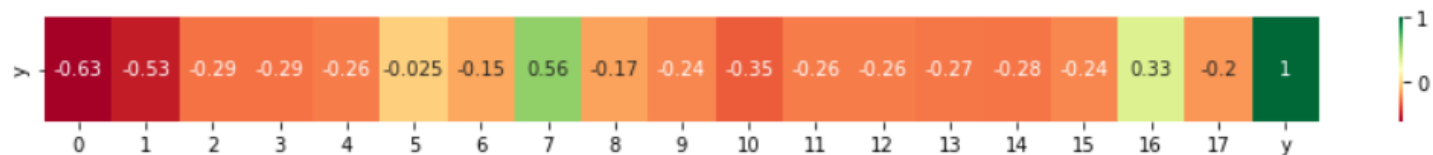
```
Out[36]: <AxesSubplot:xlabel='0', ylabel='y'>
```



```
In [39]: df.corr()['y']
```

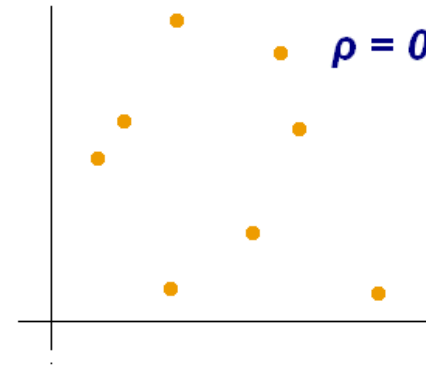
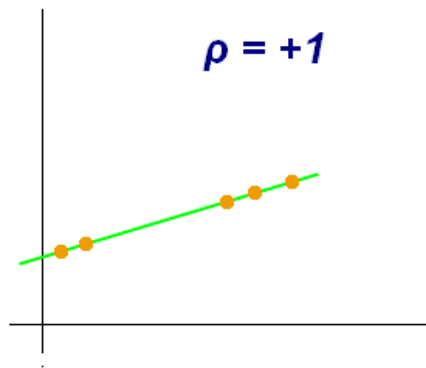
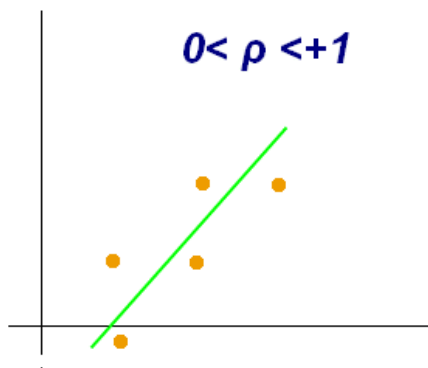
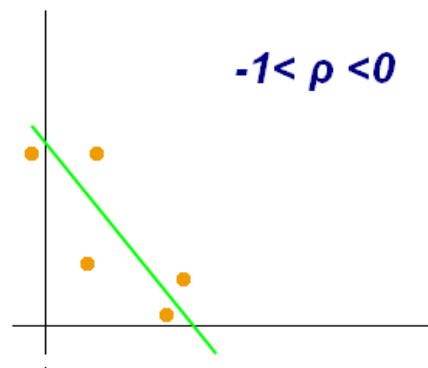
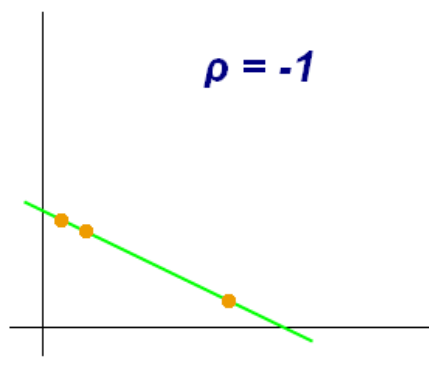
```
Out[39]: 0    -0.628485
         1    -0.528149
         2    -0.291100
         3    -0.286463
         4    -0.262104
         5    -0.024741
         6    -0.152404
         7     0.564631
         8    -0.169864
         9    -0.236039
        10    -0.347411
        11    -0.264994
        12    -0.260405
        13    -0.274515
        14    -0.279717
        15    -0.240431
        16     0.329626
        17    -0.196517
         y     1.000000
         Name: y, dtype: float64
```

```
In [40]: import matplotlib.pyplot as plt
         import seaborn as sns
         plt.figure(figsize=(15,1))
         sns.heatmap(df.corr()['y'].to_frame().T, cmap='RdYlGn', annot=True)
         plt.show()
```



Pearson相关系数

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}}$$



数据可视化

```
In [44]: plt.figure(figsize=(20,20))  
sns.heatmap(df.corr(), cmap='RdYlGn', annot=True)
```

Out[44]: <AxesSubplot:>



数据可视化

```
In [46]: import plotly.express as px  
fig = px.scatter_3d(df, x=0, y=17, z='y', color = y)  
fig.show()
```

