◆上 机 操 作

[1]:
```
#coding: utf-8
```

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

#读取Excel数据
filefullpath = r"D:/ML-Data/Paper One-1.xlsx"
data = pd.read_excel(filefullpath)
df = data.loc[:, ['ELEMENTS','T_AVE','RH_AVE', 'TOW', 'PRECIPIT', 'SOLAR', 'ULTRA','Cl', 'SO2', 'Vcorr']].copy()
#iloc它接受的是一个数字，代表着要选择数据的位置；loc接受标签的索引
#读取数据特征
df.describe()
```

[2]:

	ELEMENTS	T_AVE	RH_AVE	TOW	PRECIPIT	SOLAR	ULTRA	Cl	SO2	Vcorr
count	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000
mean	2.386491	17.963636	77.590909	4754.181818	1600.827273	5489.584848	311.287879	31.039394	3.659091	0.058317
std	2.050521	4.235551	1.634082	400.034734	364.814938	689.295426	37.026974	21.110949	1.238320	0.043650
min	0.025300	14.200000	74.000000	3992.600000	1166.400000	4193.300000	246.900000	2.050000	1.800000	0.004300
25%	1.097200	14.500000	77.000000	4498.700000	1344.500000	5216.100000	294.500000	3.300000	2.600000	0.021638
50%	1.995300	15.600000	78.000000	4748.300000	1410.800000	5517.200000	301.700000	42.100000	3.650000	0.048400
75%	2.391600	23.800000	79.000000	5092.700000	1774.800000	5744.000000	338.100000	46.250000	4.600000	0.080134
max	9.198800	24.000000	79.000000	5248.600000	2314.000000	6618.500000	372.800000	55.400000	5.300000	0.199500

[3]:
```
df.shape
```

[3]:
```
(99, 10)
```

归一化

[4]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
df_norm = sc.fit_transform(df)
```

[5]:
```
# 划分训练、测试数据集
from sklearn.model_selection import train_test_split
TrainData, TestData = train_test_split(df_norm, test_size=0.2)
```

[6]:
```
#划分输入和输出数据集
mater_train = TrainData[:, 0:9].copy()
corr_train = TrainData[:, 9].copy()
mater_test = TestData[:, 0:9].copy()
corr_test = TestData[:, 9].copy()
```

PCA

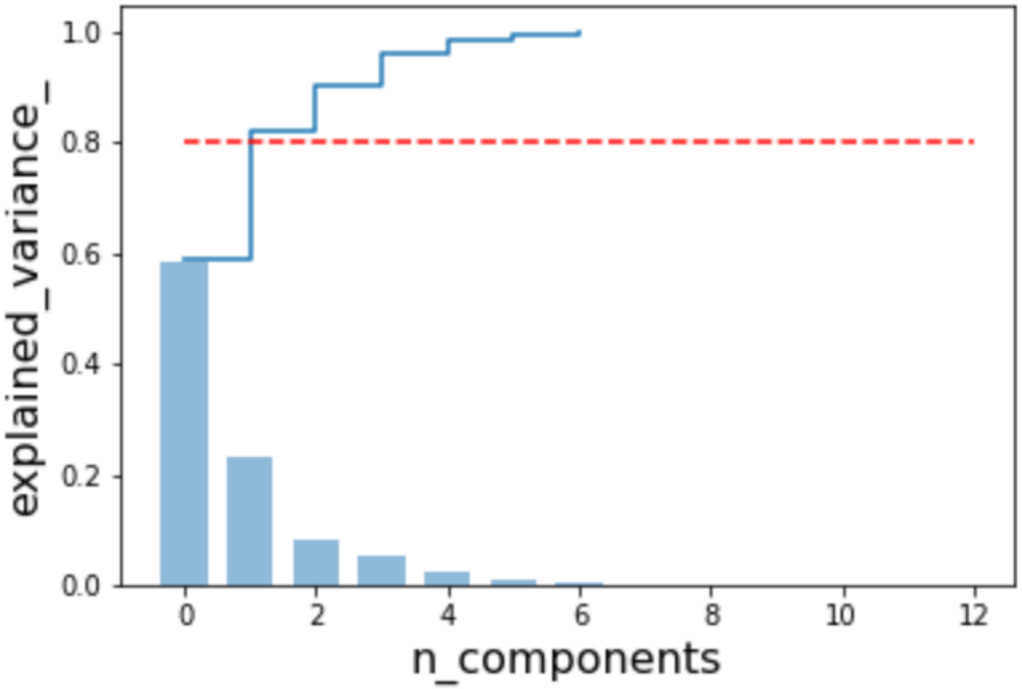
[7]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=7, whiten=True)
mater_train_pca = pca.fit_transform(mater_train)
```

[8]:
```
print(pca.explained_variance_ratio_)
#方差解释率，也就是每个特征值占所有特征值和的比例。
```

```
[0.58773283 0.23257465 0.0825087  0.0564142  0.02558503 0.01027771
 0.00381168]
```

[9]:
```
tot = sum(pca.explained_variance_)
# 求出特征值的和
var_exp = [(i / tot) for i in sorted(pca.explained_variance_, reverse=True)]
# 求出每个特征值占的比例（降序）
cum_var_exp = np.cumsum(var_exp)
# 返回var_exp的累积和
```

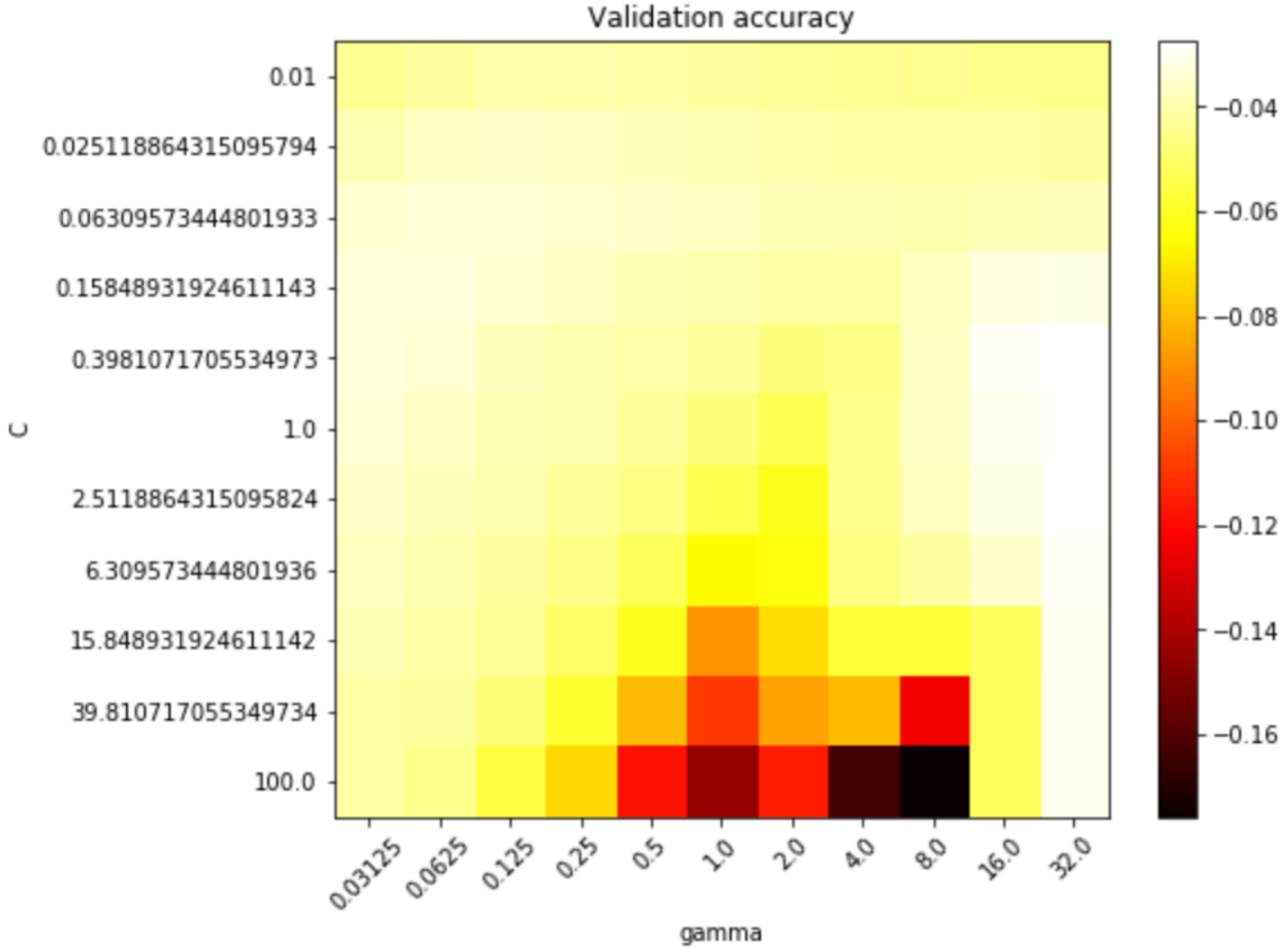
[10]:
```
plt.figure()
plt.bar(range(len(pca.explained_variance_)), pca.explained_variance_, width=0.7, bottom=0.0, alpha=0.5, label='individual explained variance')
plt.step(range(len(pca.explained_variance_)), cum_var_exp, where='post', label='cumulative explained variance')
plt.plot([0, 12], [0.8, 0.8], "r--")
plt.xlabel('n_components', fontsize=16)
plt.ylabel('explained_variance_', fontsize=16)
plt.show()
```


explained_variance_
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
n_components

SVR参数寻优

[11]:
```
#第一轮参数寻优
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
C_range = np.logspace(-2, 2, 11)
gamma_range = np.logspace(-5, 5, 11, base=2)
svr = GridSearchCV(SVR(kernel='rbf'), param_grid={'C': C_range, 'gamma': gamma_range}, scoring='neg_mean_squared_error', cv=10)
svr.fit(mater_train_pca, corr_train)
#绘制热度图
plt.figure(figsize=(8, 6))
plt.subplots_adjust(left=.2, right=0.95, bottom=0.15, top=0.95)
scores = svr.cv_results_['mean_test_score'].reshape(len(C_range), len(gamma_range))
plt.imshow(scores, interpolation='nearest', cmap=plt.cm.hot)
plt.xlabel('gamma')
plt.ylabel('C')
plt.colorbar()
plt.xticks(np.arange(len(gamma_range)), gamma_range, rotation=45)
plt.yticks(np.arange(len(C_range)), C_range)
plt.title('Validation accuracy')
plt.show()
```

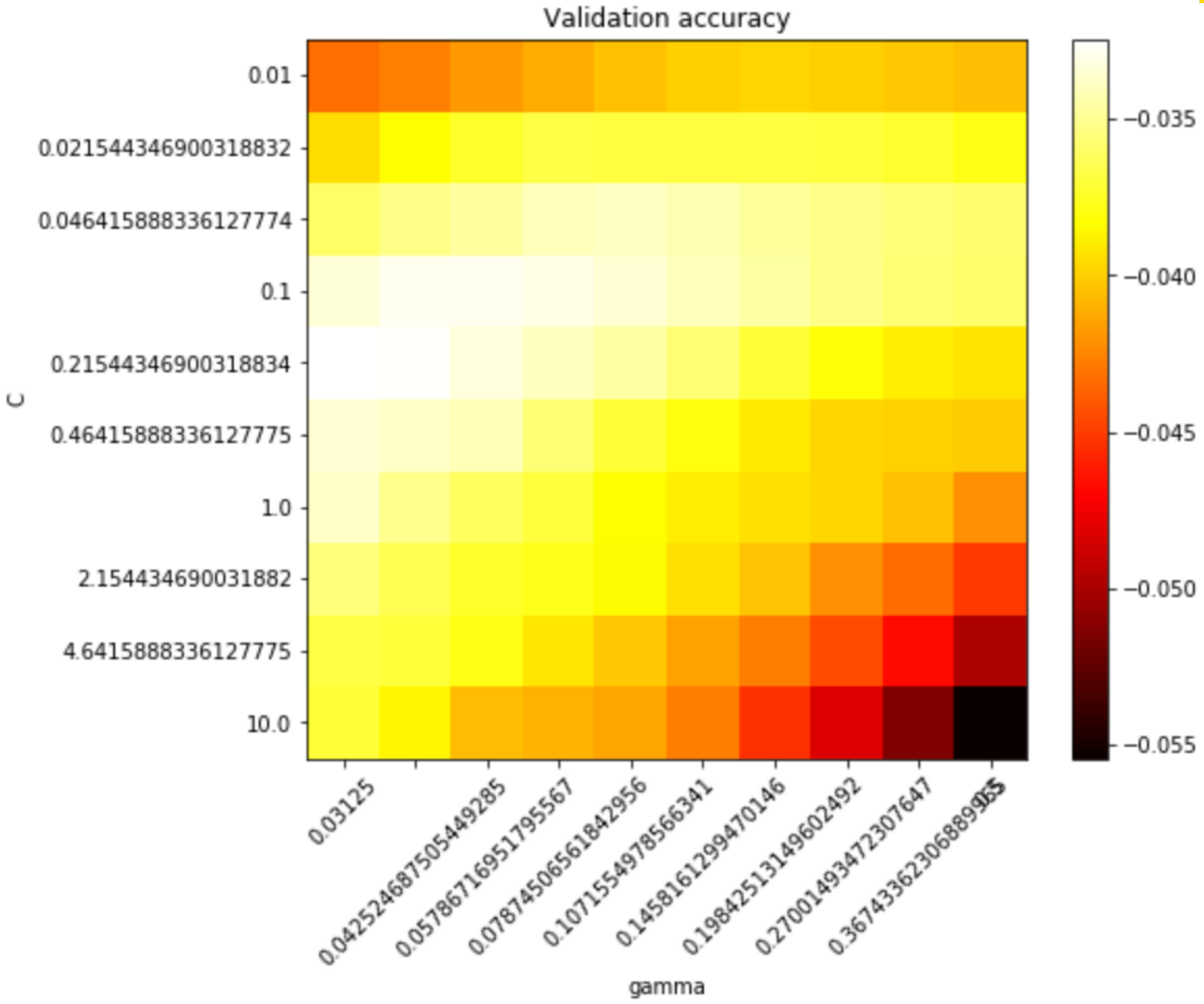
[12]:
```
print (svr.best_params_)
```




```
{'C': 0.3981071705534973, 'gamma': 32.0}
```

3. 机器学习算法

[13]:
```
#第二轮参数寻优
C_range = np.logspace(-2, 1, 10)
gamma_range = np.logspace(-5, -1, 10, base=2)
svr = GridSearchCV(SVR(kernel='rbf'), param_grid={'C': C_range, 'gamma': gamma_range}, scoring='neg_mean_squared_error', cv=10)
svr.fit(mater_train_pca, corr_train)
#绘制热度图
plt.figure(figsize=(8, 6))
plt.subplots_adjust(left=.2, right=0.95, bottom=0.15, top=0.95)
scores = svr.cv_results_['mean_test_score'].reshape(len(C_range), len(gamma_range))
plt.imshow(scores, interpolation='nearest', cmap=plt.cm.hot)
plt.xlabel('gamma')
plt.ylabel('C')
plt.colorbar()
plt.xticks(np.arange(len(gamma_range)), gamma_range, rotation=45)
plt.yticks(np.arange(len(C_range)), C_range)
plt.title('Validation accuracy')
plt.show()
```


Validation accuracy
C
0.01
0.021544346900318832
0.046415888336127774
0.1
0.21544346900318834
0.46415888336127775
1.0
2.154434690031882
4.6415888336127775
10.0
gamma
0.03125
0.042524687505449285
0.057867169517955 67
0.078745065618429 56
0.107155497856634 1
0.145816129947014 6
0.198425131496024 92
0.270014934723076 47
0.367433623068899 65
−0.035
−0.040
−0.045
−0.050
−0.055

[14]:
```
print (svr.best_params_)
```

```
{'C': 0.21544346900318834, 'gamma': 0.03125}
```

[15]:
```
model = svr.best_estimator_
```

训练集

[16]:
```
result_train = model.predict(mater_train_pca) #预测结果
#为进行反归一化，将输入和输出数据合并
INVERSE_TrainData = np.column_stack((mater_train, result_train))
Inversed_TrainData = sc.inverse_transform(INVERSE_TrainData)
corr_train_fit = Inversed_TrainData[:, 9].copy()
TrainData_unnorm = sc.inverse_transform(TrainData)#前面是先做的归一化再做的拆分
corr_train_mea = TrainData_unnorm[:, 9].copy()
```

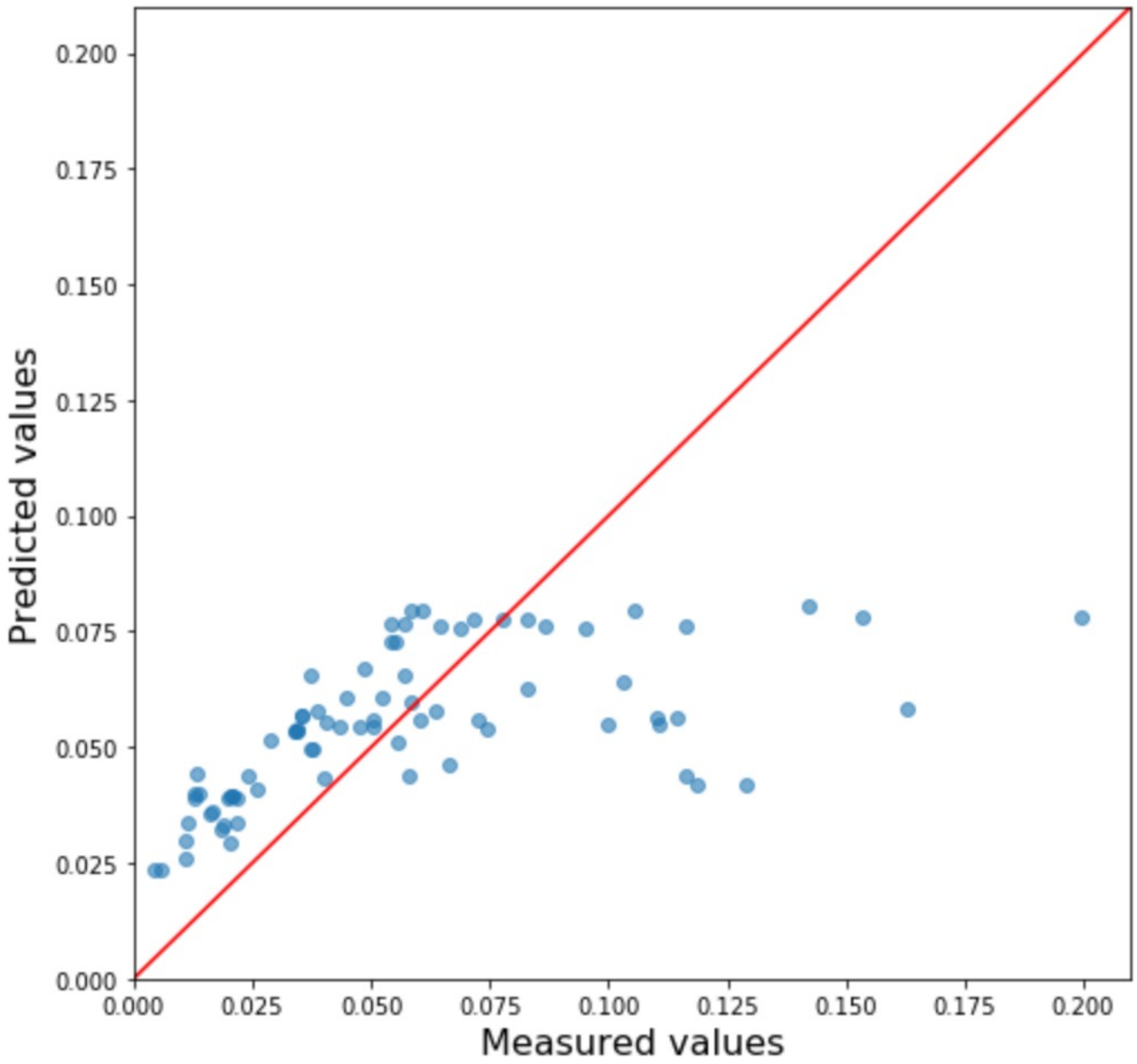
[17]:
```
#均方误差（Mean squared error）
from sklearn.metrics import mean_squared_error
mean_squared_error(corr_train_mea, corr_train_fit)
```

[17]:
```
0.0011150558756451354
```

[18]:
```
#求R方值
from sklearn.metrics import r2_score
r2_score(corr_train_mea, corr_train_fit)
```

[18]:
```
0.31159148434295025
```

[19]:
```
plt.figure(figsize=(8, 8))
plt.scatter(corr_train_mea, corr_train_fit, alpha=0.6)
plt.plot([0, 0.21], [0, 0.21], "r-")
plt.xlabel('Measured values', fontsize=16)
plt.ylabel('Predicted values', fontsize=16)
plt.xlim(0, 0.21)
plt.ylim(0, 0.21)
plt.show()
```


Predicted values
Measured values
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200

测试集

[20]:
```
mater_test_pca = pca.transform(mater_test)
result_test = model.predict(mater_test_pca) #求得测试集样品的腐蚀速率预测值
#反归一化
INVERSE_TestData2 = np.column_stack((mater_test, result_test))
Inversed_TestData2 = sc.inverse_transform(INVERSE_TestData2)
corr_test_fit = Inversed_TestData2[:, 9].copy()
TestData_unnorm = sc.inverse_transform(TestData)#前面是先做的归一化再做的拆分
corr_test_mea = TestData_unnorm[:, 9].copy()
```

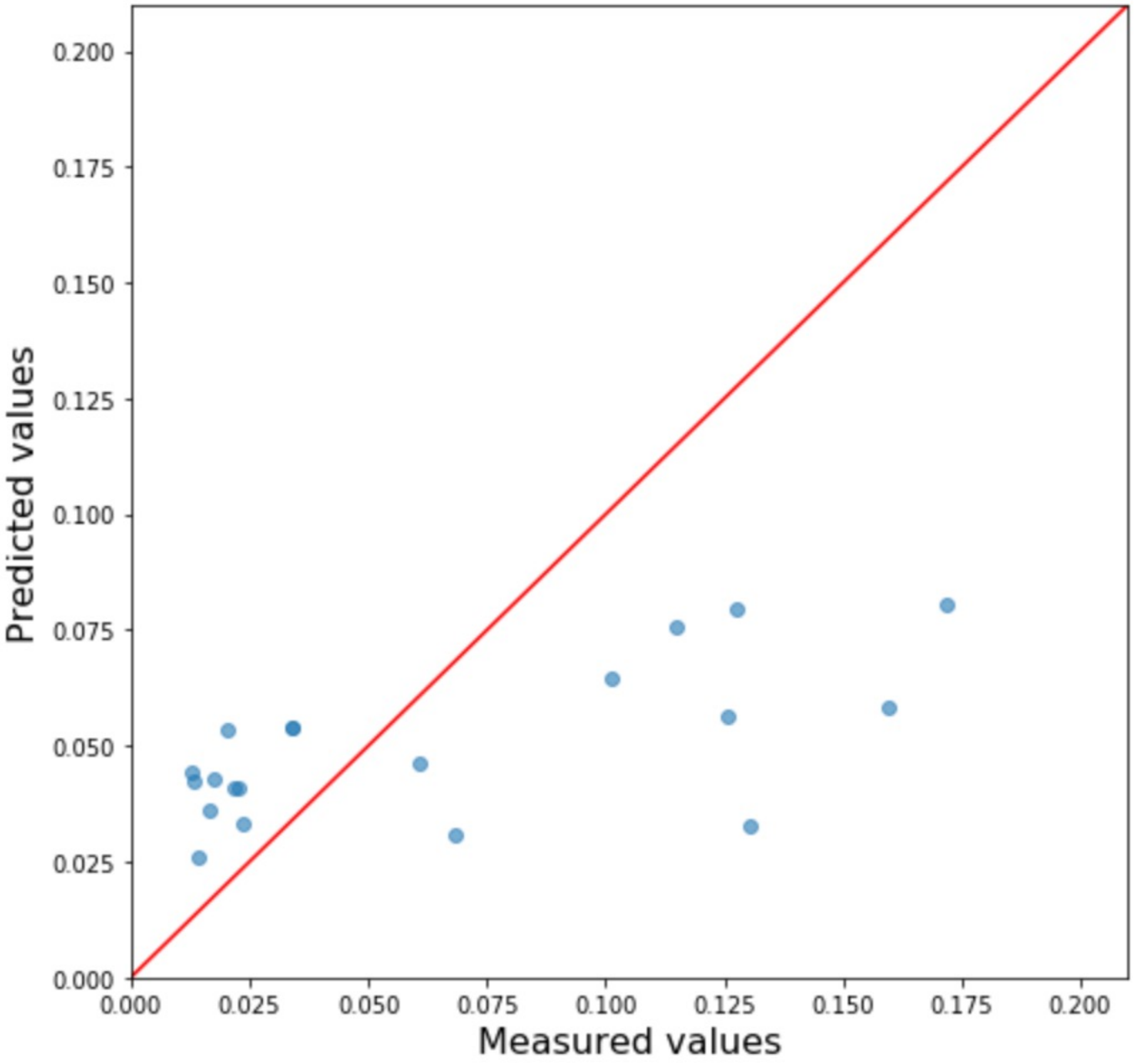
[21]:
```
#均方误差（Mean squared error）
mean_squared_error(corr_test_mea, corr_test_fit)
```

[21]:
```
0.0022658940388209175
```

[22]:
```
#求R方值
r2_score(corr_test_mea, corr_test_fit)
```

[22]:
```
0.21574675277236754
```

[23]:
```
plt.figure(figsize=(8, 8))
plt.scatter(corr_test_mea, corr_test_fit, alpha=0.6)
plt.plot([0, 0.21], [0, 0.21], "r-")
plt.xlabel('Measured values', fontsize=16)
plt.ylabel('Predicted values', fontsize=16)
plt.xlim(0, 0.21)
plt.ylim(0, 0.21)
plt.show()
```


Predicted values
Measured values
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200