# ◆上 机 操 作

## ◆ 小练习

[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"G:/DATA/Paper One-ForVisual.xlsx"
df = pd.read_excel(filefullpath)
#读取数据特征
df.describe()
```

[2]:

| | ELEMENTS | T_MAX | T_MIN | T_AVE | RH_MIN | RH |
|---|---|---|---|---|---|---|
| count | 522.000000 | 522.000000 | 522.000000 | 522.000000 | 522.000000 | 522.0 |
| mean | 2.367471 | 34.321839 | -0.923563 | 17.961494 | 39.214943 | 77.1 |
| std | 1.991431 | 1.743940 | 6.744028 | 4.199953 | 11.876820 | 1.9 |
| min | 0.030000 | 31.300000 | -8.000000 | 14.200000 | 16.800000 | 72.8 |

# 3. 机器学习算法

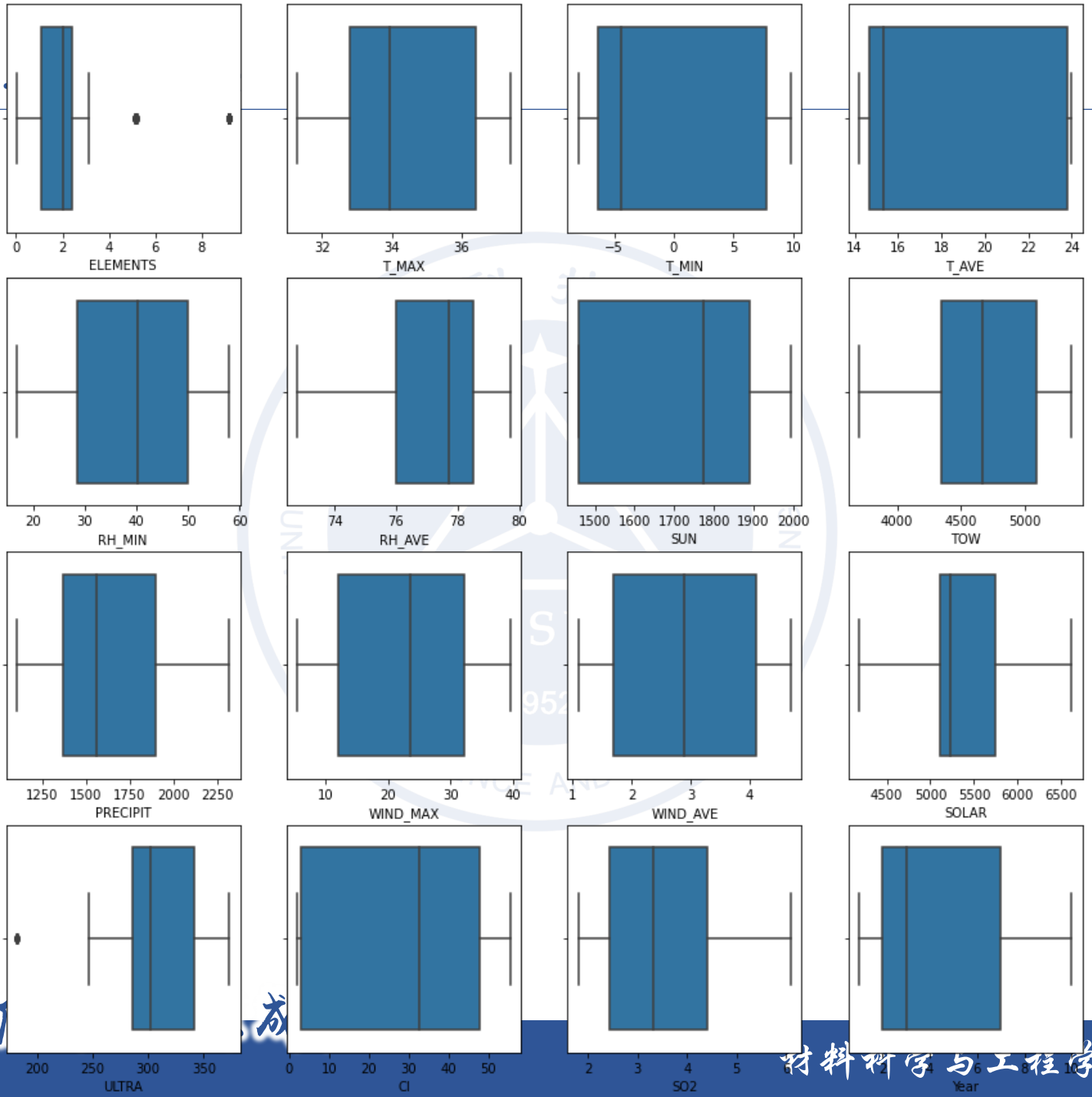
[4]:
```
#检查重复值
print('未去重: ', df.shape)
print('去重: ', df.drop_duplicates().shape)
```

```
未去重:  (522, 17)
去重:  (511, 17)
```

[5]:
```
df.columns
```

[5]:
```
Index(['ELEMENTS', 'T_MAX', 'T_MIN', 'T_AVE', 'RH_MIN', 'RH_AVE', 'SUN', 'TOW',
       'PRECIPIT', 'WIND_MAX', 'WIND_AVE', 'SOLAR', 'ULTRA', 'Cl', 'SO2',
       'Year', 'Vcorr'],
      dtype='object')
```

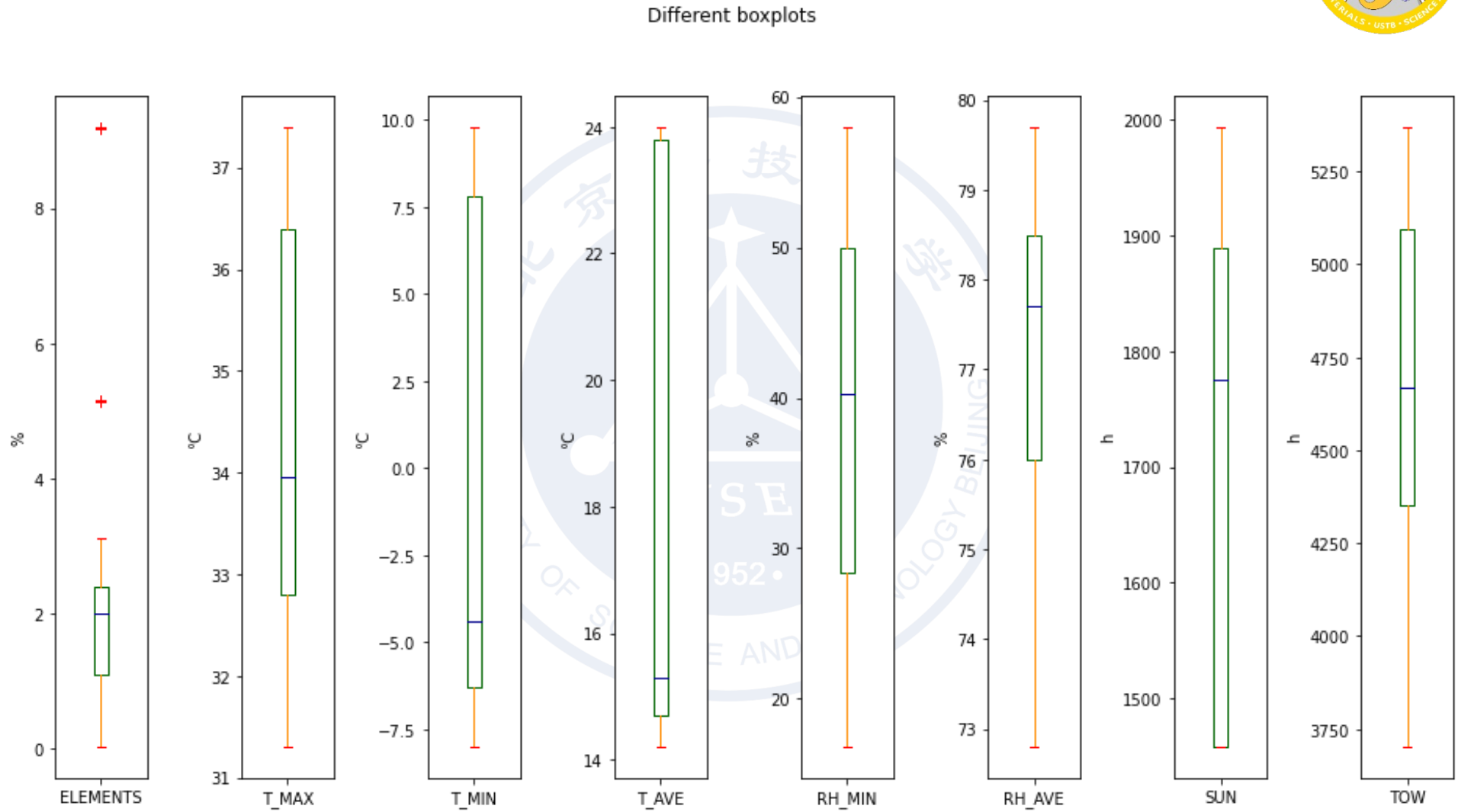
[6]:
```
#检查异常值
#箱线图
fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(15, 15))
#绘制箱线图
sns.boxplot(x="ELEMENTS", data=df,ax=axes[0][0])
sns.boxplot(x='T_MAX', data=df, ax=axes[0][1])
sns.boxplot(x='T_MIN', data=df, ax=axes[0][2])
sns.boxplot(x='T_AVE', data=df, ax=axes[0][3])
sns.boxplot(x='RH_MIN', data=df, ax=axes[1][0])
sns.boxplot(x="RH_AVE", data=df,ax=axes[1][1])
sns.boxplot(x='SUN', data=df, ax=axes[1][2])
sns.boxplot(x='TOW', data=df, ax=axes[1][3])
sns.boxplot(x='PRECIPIT', data=df, ax=axes[2][0])
sns.boxplot(x='WIND_MAX', data=df, ax=axes[2][1])
sns.boxplot(x='WIND_AVE', data=df, ax=axes[2][2])
sns.boxplot(x='SOLAR', data=df, ax=axes[2][3])
sns.boxplot(x='ULTRA', data=df, ax=axes[3][0])
sns.boxplot(x='Cl', data=df, ax=axes[3][1])
sns.boxplot(x='SO2', data=df, ax=axes[3][2])
sns.boxplot(x='Year', data=df, ax=axes[3][3])
plt.show()
```


ELEMENTS
T_MAX
T_MIN
T_AVE
RH_MIN
RH_AVE
SUN
TOW
PRECIPIT
WIND_MAX
WIND_AVE
SOLAR
ULTRA
CI
SO2
Year

## ◆ 另一种箱线图

```
7]: df1 = df.iloc[:, 0:8].copy()
    df2 = df.iloc[:, 8:17].copy()
    #划分
    fig, axes = plt.subplots(1,8,figsize=(16,8)) #这个函数是设置子图参数的
    color = dict(boxes='DarkGreen', whiskers='DarkOrange', medians='DarkBlue', caps='Red')
    # boxes表示箱体，whisker表示触须线
    # medians表示中位数，caps表示最大与最小值界限
    df1.plot(kind='box',ax=axes, subplots=True, title='Different boxplots', color=color, sym='r+')
    # sym参数表示异常值标记的方式
    axes[0].set_ylabel('%')
    axes[1].set_ylabel('℃')
    axes[2].set_ylabel('℃')
    axes[3].set_ylabel('℃')
    axes[4].set_ylabel('%')
    axes[5].set_ylabel('%')
    axes[6].set_ylabel('h')
    axes[7].set_ylabel('h')
    fig.subplots_adjust(wspace=1,hspace=1)  # 调整子图之间的间距
    plt.show()
```

# 3. 机器学习算法

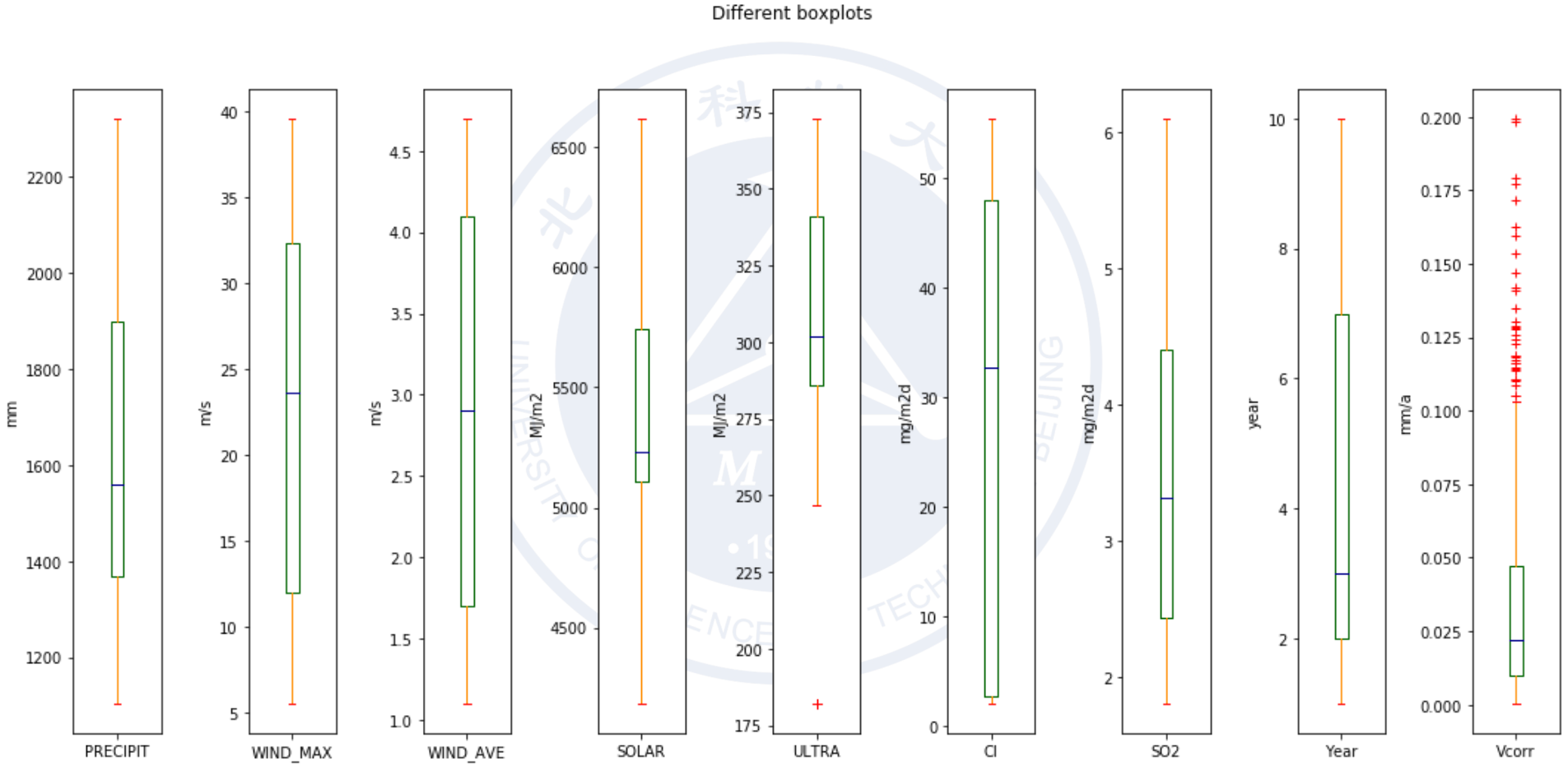

Different boxplots
%
ELEMENTS
°C
T_MAX
°C
T_MIN
°C
T_AVE
%
RH_MIN
%
RH_AVE
h
SUN
h
TOW

◆绘制出变量 ‘PRECIPIT’, ‘WIND_MAX’,

‘WIND_AVE’, ‘SOLAR’, ‘ULTRA’, ‘Cl’,

‘SO2’, ‘Year’, ‘Vcorr‘的箱线图

# 3. 机器学习算法


Different boxplots
mm
PRECIPIT
m/s
WIND_MAX
m/s
WIND_AVE
MJ/m2
SOLAR
MJ/m2
ULTRA
mg/m2d
Cl
mg/m2d
SO2
year
Year
mm/a
Vcorr

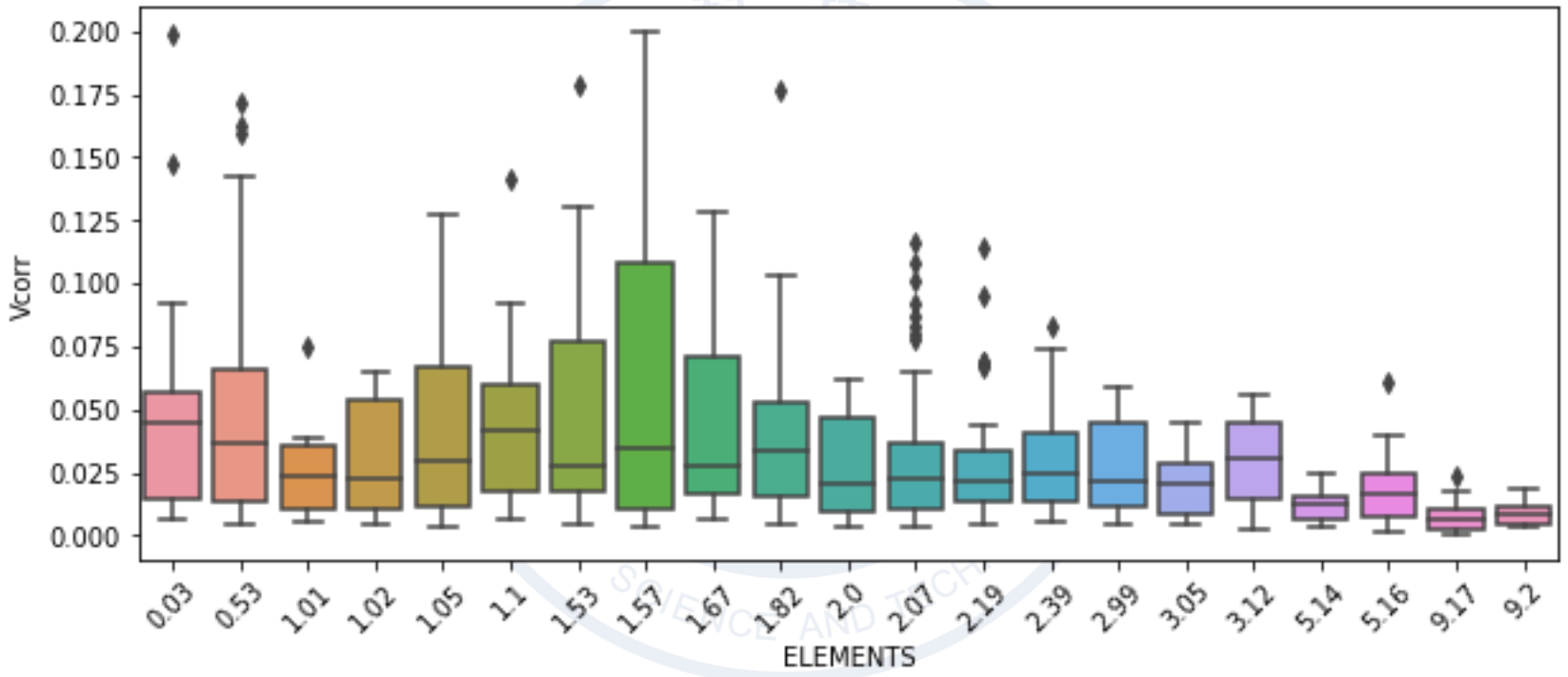
## ◆ 上图的代码

```
fig, axes = plt.subplots(1,9,figsize=(18,8)) #这个函数是设置子图参数的
color = dict(boxes='DarkGreen', whiskers='DarkOrange', medians='DarkBlue', caps='Red')
# boxes表示箱体，whisker表示触须线
# medians表示中位数，caps表示最大与最小值界限
df2.plot(kind='box',ax=axes, subplots=True, title='Different boxplots', color=color, sym='r+')
# sym参数表示异常值标记的方式
axes[0].set_ylabel('mm')
axes[1].set_ylabel('m/s')
axes[2].set_ylabel('m/s')
axes[3].set_ylabel('MJ/m2')
axes[4].set_ylabel('MJ/m2')
axes[5].set_ylabel('mg/m2d')
axes[6].set_ylabel('mg/m2d')
axes[7].set_ylabel('year')
axes[8].set_ylabel('mm/a')
fig.subplots_adjust(wspace=1,hspace=1)  # 调整子图之间的间距
plt.show()
```

◆ 数据分布的展示分析

## 可视化分析

[8]:
```
#1）合金化程度与腐蚀速率
plt.figure(figsize=(10, 4))
plt.xticks(rotation=45)
sns.boxplot(x='ELEMENTS', y='Vcorr', data=df)
plt.show()
```


0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Vcorr
0.03
0.53
1.01
1.02
1.05
1.1
1.53
1.57
1.67
1.82
2.0
2.07
2.19
2.39
2.99
3.05
3.12
5.14
5.16
9.17
9.2
ELEMENTS

◆绘制出变量 ‘T_MAX’, ‘T_MIN’, ‘T_AVE’, ‘RH_MIN’,

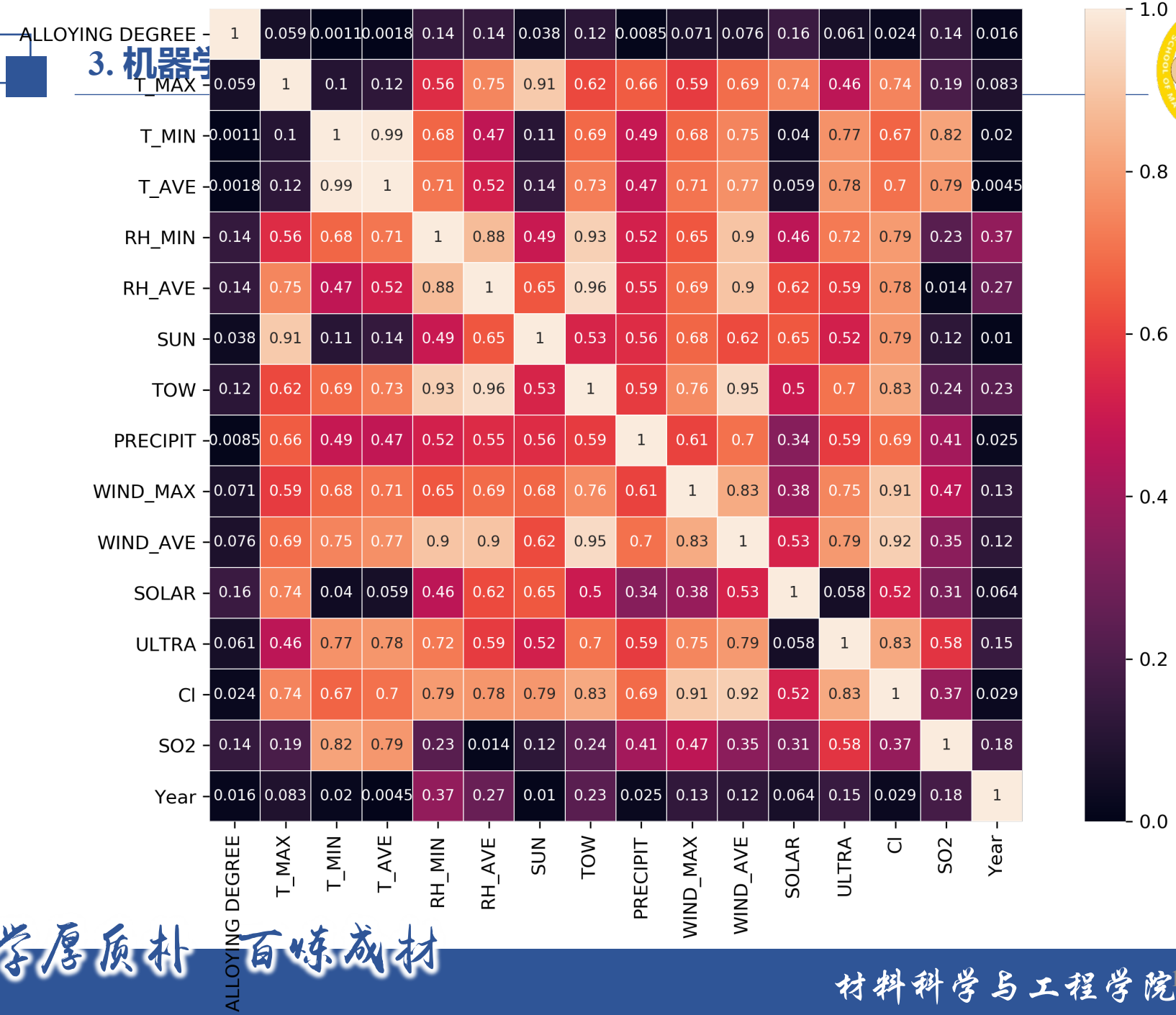
‘RH_AVE’, ‘SUN’, ‘TOW’, ‘PRECIPIT’,

‘WIND_MAX’, ‘WIND_AVE’, ‘SOLAR’, ‘ULTRA’,

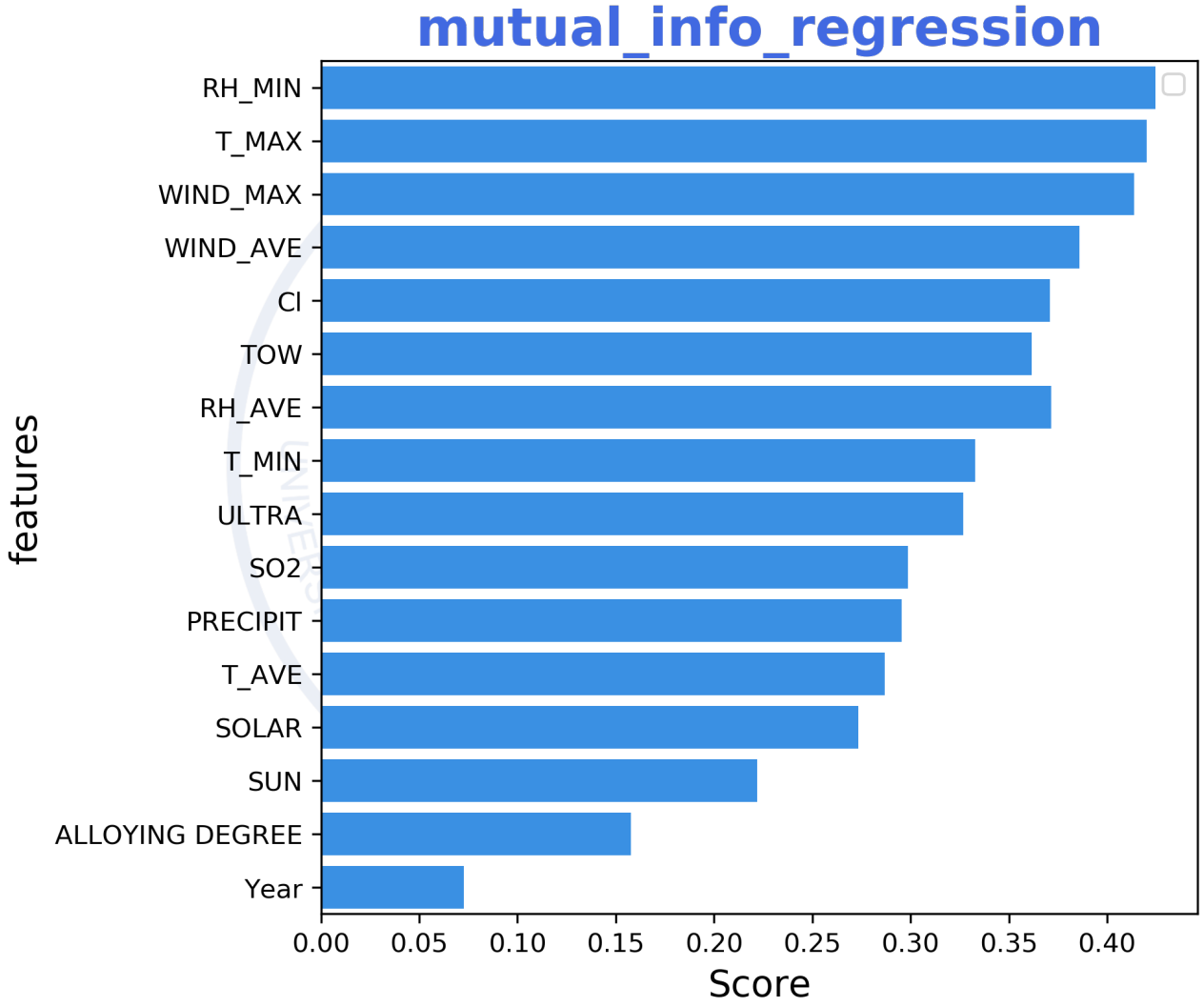
‘Cl’, ‘SO2‘, ’Year’, ‘Vcorr‘ 的数值分布图

## ◆ 特征工程

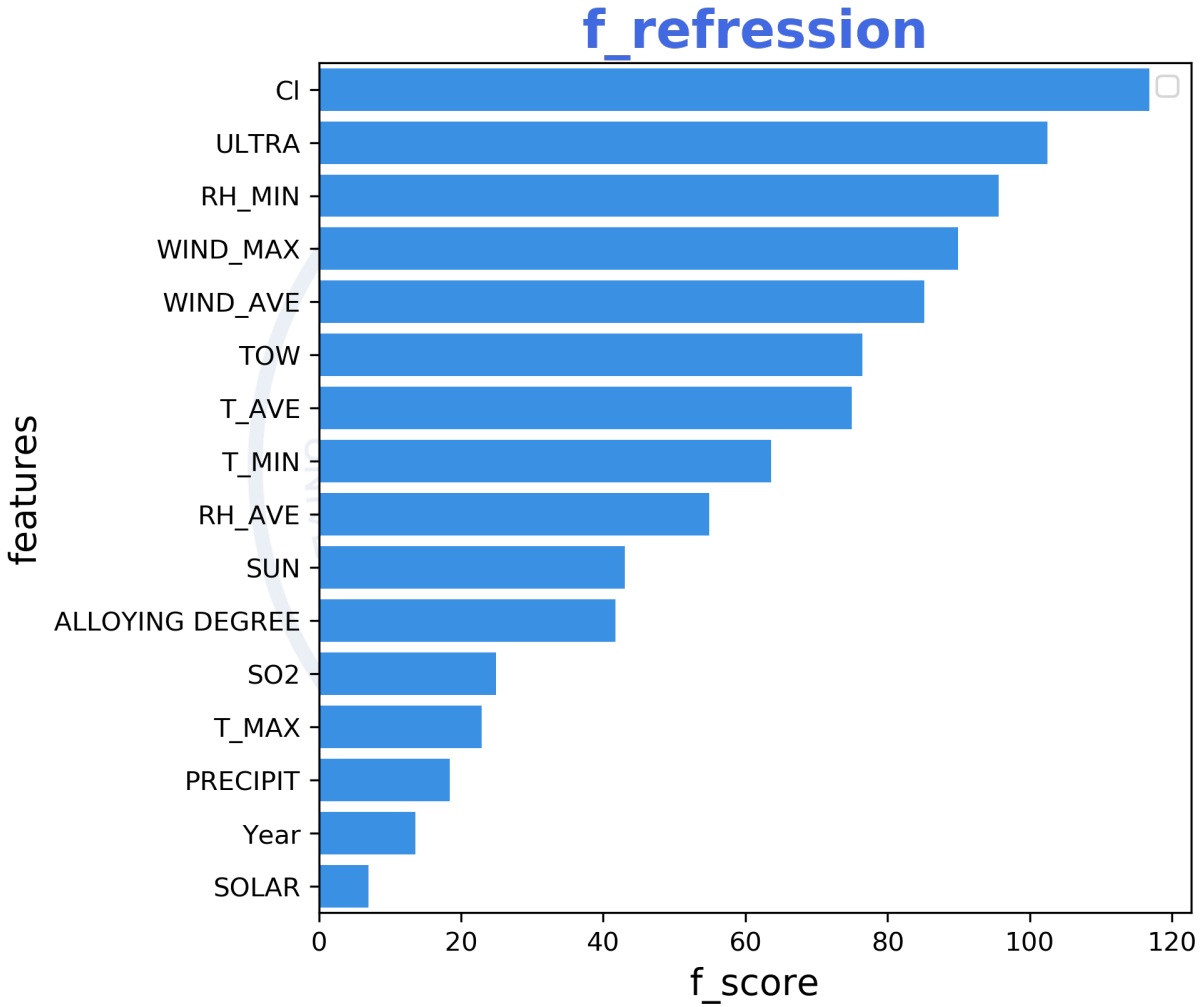
```
#归一化
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
Data_Nor = sc.fit_transform(df)
mater_Data2 = Data_Nor[:, 0:16].copy()  #除去了Vcorr那一列
mater_Data2 = pd.DataFrame(mater_Data2) #归一化后矩阵变成np.array格式，不能做.corr()，所以要转换成dataframe
mater_Data2.columns = ['ALLOYING DEGREE', 'T_MAX', 'T_MIN', 'T_AVE', 'RH_MIN', 'RH_AVE', 'SUN',
       'TOW', 'PRECIPIT', 'WIND_MAX', 'WIND_AVE', 'SOLAR', 'ULTRA', 'Cl',
       'SO2', 'Year'] #转换后的dataframe列名称都丢了，重新赋值，否则图里的特征名称都是0.1.2这样
correlation2 = mater_Data2.corr(method='pearson')
correlation2_abs = correlation2.abs()
plt.figure(figsize=(12, 8),dpi=300)
#绘制热力图
sns.heatmap(correlation2_abs, linewidths=0.2, vmax=1, vmin=0, linecolor='w',
            annot=True,annot_kws={'size':8},square=True)
```


ALLOYING DEGREE 1 0.059 0.0011 0.0018 0.14 0.14 0.038 0.12 0.0085 0.071 0.076 0.16 0.061 0.024 0.14 0.016
T_MAX 0.059 1 0.1 0.12 0.56 0.75 0.91 0.62 0.66 0.59 0.69 0.74 0.46 0.74 0.19 0.083
T_MIN 0.0011 0.1 1 0.99 0.68 0.47 0.11 0.69 0.49 0.68 0.75 0.04 0.77 0.67 0.82 0.02
T_AVE 0.0018 0.12 0.99 1 0.71 0.52 0.14 0.73 0.47 0.71 0.77 0.059 0.78 0.7 0.79 0.0045
RH_MIN 0.14 0.56 0.68 0.71 1 0.88 0.49 0.93 0.52 0.65 0.9 0.46 0.72 0.79 0.23 0.37
RH_AVE 0.14 0.75 0.47 0.52 0.88 1 0.65 0.96 0.55 0.69 0.9 0.62 0.59 0.78 0.014 0.27
SUN 0.038 0.91 0.11 0.14 0.49 0.65 1 0.53 0.56 0.68 0.62 0.65 0.52 0.79 0.12 0.01
TOW 0.12 0.62 0.69 0.73 0.93 0.96 0.53 1 0.59 0.76 0.95 0.5 0.7 0.83 0.24 0.23
PRECIPIT 0.0085 0.66 0.49 0.47 0.52 0.55 0.56 0.59 1 0.61 0.7 0.34 0.59 0.69 0.41 0.025
WIND_MAX 0.071 0.59 0.68 0.71 0.65 0.69 0.68 0.76 0.61 1 0.83 0.38 0.75 0.91 0.47 0.13
WIND_AVE 0.076 0.69 0.75 0.77 0.9 0.9 0.62 0.95 0.7 0.83 1 0.53 0.79 0.92 0.35 0.12
SOLAR 0.16 0.74 0.04 0.059 0.46 0.62 0.65 0.5 0.34 0.38 0.53 1 0.058 0.52 0.31 0.064
ULTRA 0.061 0.46 0.77 0.78 0.72 0.59 0.52 0.7 0.59 0.75 0.79 0.058 1 0.83 0.58 0.15
CI 0.024 0.74 0.67 0.7 0.79 0.78 0.79 0.83 0.69 0.91 0.92 0.52 0.83 1 0.37 0.029
SO2 0.14 0.19 0.82 0.79 0.23 0.014 0.12 0.24 0.41 0.47 0.35 0.31 0.58 0.37 1 0.18
Year 0.016 0.083 0.02 0.0045 0.37 0.27 0.01 0.23 0.025 0.13 0.12 0.064 0.15 0.029 0.18 1
ALLOYING DEGREE
T_MAX
T_MIN
T_AVE
RH_MIN
RH_AVE
SUN
TOW
PRECIPIT
WIND_MAX
WIND_AVE
SOLAR
ULTRA
CI
SO2
Year
1.0
0.8
0.6
0.4
0.2
0.0

```
from sklearn.feature_selection import mutual_info_regression
features = mater_Data2
target = Data_Nor[:, 16].copy()
score = mutual_info_regression(features, target)
X = ['ALLOYING DEGREE', 'T_MAX', 'T_MIN', 'T_AVE', 'RH_MIN', 'RH_AVE', 'SUN',
       'TOW', 'PRECIPIT', 'WIND_MAX', 'WIND_AVE', 'SOLAR', 'ULTRA', 'Cl',
       'SO2', 'Year']
y = score
plt.figure(figsize=(6, 6),dpi=300)
sns.barplot(y, X, color="dodgerblue", order=[ 'RH_MIN','T_MAX','WIND_MAX', 'WIND_AVE'
        'SO2', 'PRECIPIT','T_AVE', 'SOLAR','SUN', 'ALLOYING DEGREE','Year'])
plt.xlabel('Score',fontsize=14)
plt.ylabel('features',fontsize=14)
plt.legend()
plt.title('mutual_info_regression',fontsize=20,fontweight='bold',color='royalblue')
plt.show()
```


mutual_info_regression
RH_MIN
T_MAX
WIND_MAX
WIND_AVE
CI
TOW
RH_AVE
T_MIN
ULTRA
SO2
PRECIPIT
T_AVE
SOLAR
SUN
ALLOYING DEGREE
Year
features
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Score

```
from sklearn.feature_selection import f_regression
score2 = f_regression(features, target)[0]
#不加最后的【0】，则会返回包含连个阵列的数值，前一个是各特征的F值，后一个是p值。
X = ['ALLOYING DEGREE', 'T_MAX', 'T_MIN', 'T_AVE', 'RH_MIN', 'RH_AVE', 'SUN',
       'TOW', 'PRECIPIT', 'WIND_MAX', 'WIND_AVE', 'SOLAR', 'ULTRA', 'Cl',
       'SO2', 'Year']
y = score2
plt.figure(figsize=(6, 6),dpi=300)
sns.barplot(y, X, color="dodgerblue",order=[ 'Cl','ULTRA','RH_MIN','WIND_MAX',
         'SUN', 'ALLOYING DEGREE','SO2', 'T_MAX','PRECIPIT','Year','SOLAR'])
plt.xlabel('f_score',fontsize=14)
plt.ylabel('features',fontsize=14)
plt.legend()
plt.title('f_refression',fontsize=20,fontweight='bold',color='royalblue')
plt.show()
```


f_refression
Cl
ULTRA
RH_MIN
WIND_MAX
WIND_AVE
TOW
T_AVE
T_MIN
RH_AVE
SUN
ALLOYING DEGREE
SO2
T_MAX
PRECIPIT
Year
SOLAR
features
0
20
40
60
80
100
120
f_score

```
X1 = df.iloc[:, 0:15].copy()
y1 = df.iloc[:, 15].copy()

from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(random_state=10)

def RFFeaImpor_(X_data,y_data):
    model.fit(X_data,y_data)
    result_ = {'var':X_data.columns.values
           ,'importances_':model.feature_importances_}
    feature_importances_ = pd.DataFrame(result_, columns=['var','importances_'], index=X
    return feature_importances_
feature_importances_ = RFFeaImpor_(X1,y1)                    index=X_data.columns.values)
print (feature_importances_)
```

| | var | importances_ |
|---|---|---|
| ELEMENTS | ELEMENTS | 0.419420 |
| T_MAX | T_MAX | 0.025188 |
| T_MIN | T_MIN | 0.014046 |
| T_AVE | T_AVE | 0.043115 |
| RH_MIN | RH_MIN | 0.063493 |
| RH_AVE | RH_AVE | 0.000468 |
| SUN | SUN | 0.062548 |
| TOW | TOW | 0.209315 |
| PRECIPIT | PRECIPIT | 0.000017 |
| WIND_MAX | WIND_MAX | 0.025196 |
| WIND_AVE | WIND_AVE | 0.011529 |
| SOLAR | SOLAR | 0.000352 |
| ULTRA | ULTRA | 0.099836 |
| Cl | Cl | 0.012945 |
| SO2 | SO2 | 0.012533 |