对于第一个问题：

1.一些基本的处理：

import pandas as pd

df=pd.DataFrame()

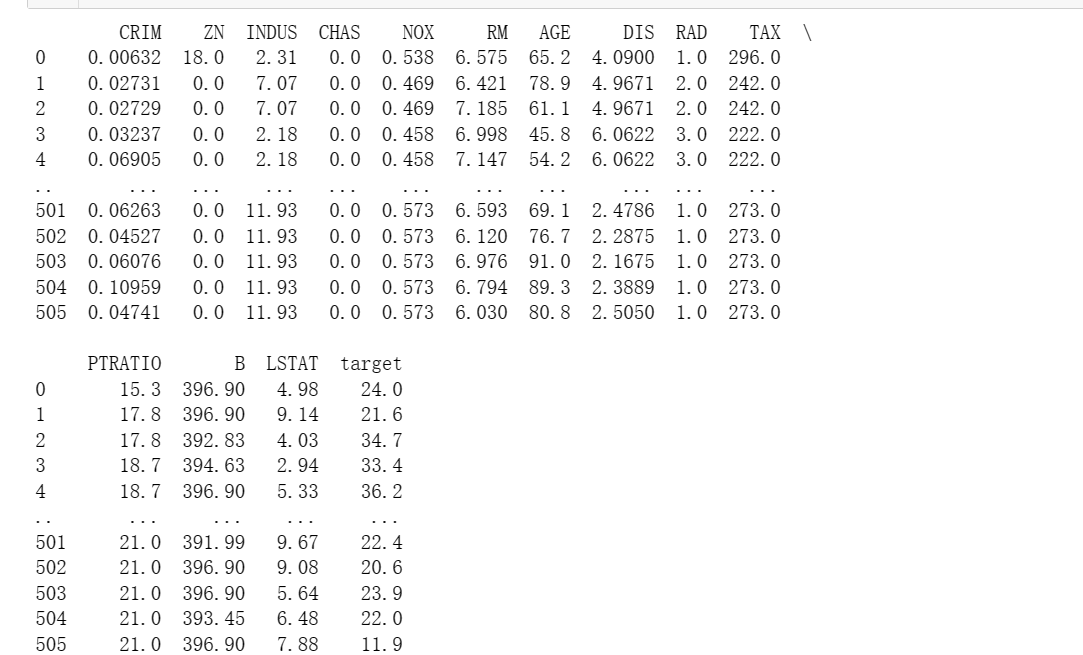
for i in range(X.shape[1]):

df[housing.feature\_names[i]]=X[:,i]

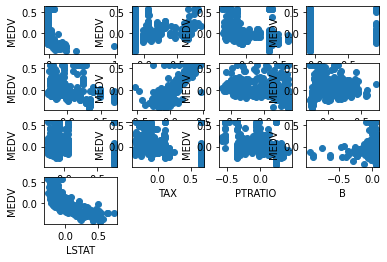
df['target']=y

df.to\_csv('boston\_housing.csv',index=None)

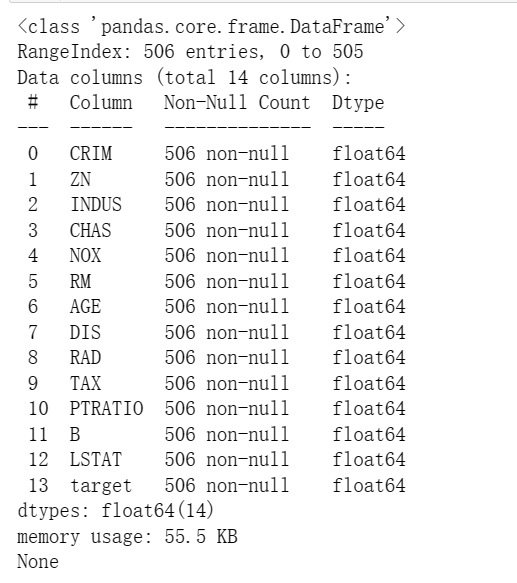
print(df)



以及把特征可视化



查看空值



很幸运地发现没有空值

plt.boxplot(X[:,0],showmeans=True,meanline=True)

plt.show()

#检测离群点

试图检查离群点，可以看出大部分特征是没有离群点的。

import pandas as pd

import matplotlib.pyplot as plt

# 读取数据

data = pd.read\_csv('D:/poston/housing.csv', delim\_whitespace=True, header=None)

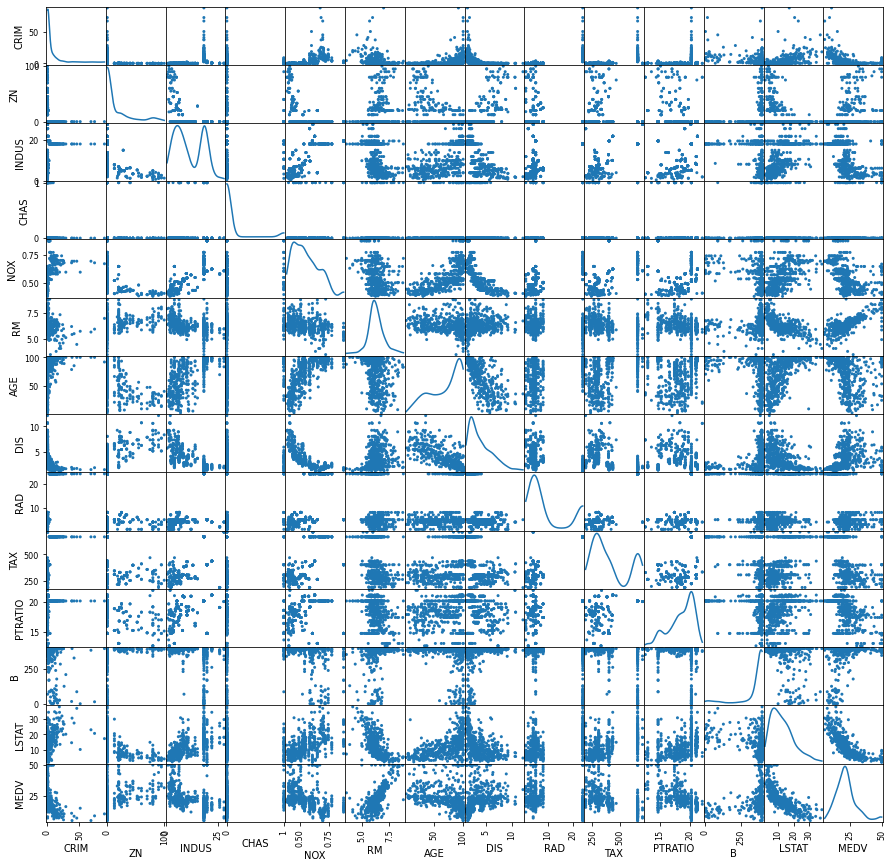
# 添加列名

data.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

# 绘制矩阵图

pd.plotting.scatter\_matrix(data, alpha=1, figsize=(15, 15), diagonal='kde')

plt.show()



# 读取数据

df = pd.read\_csv('D:/poston/housing.csv', delim\_whitespace=True, header=None)

# 设置列名

feature\_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

df.columns = feature\_names

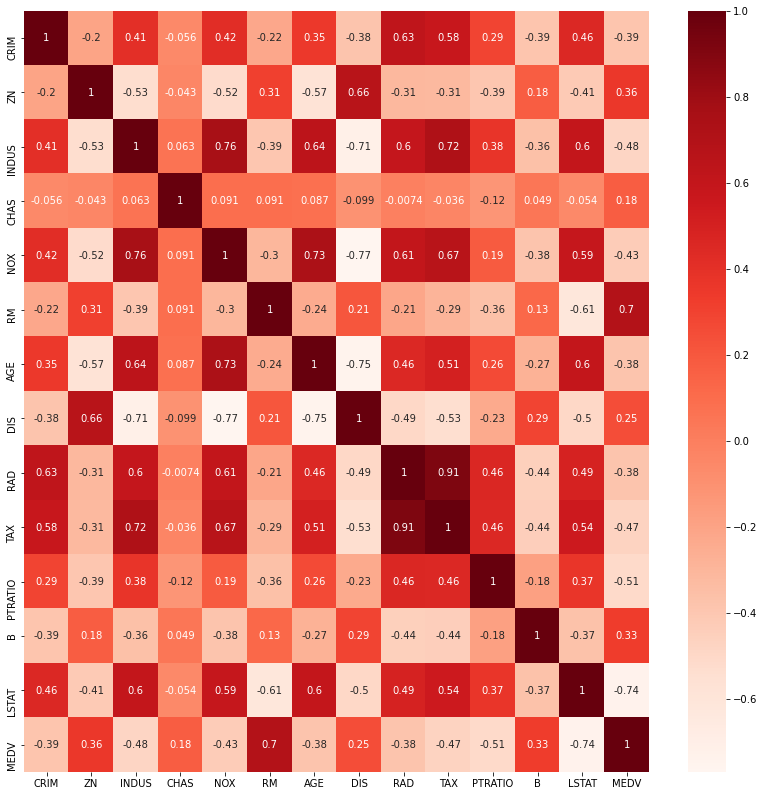
# 绘制相关系数矩阵

corr\_matrix = df.corr()

plt.figure(figsize=(14, 14))

sns.heatmap(corr\_matrix, annot=True, cmap=plt.cm.Reds)

plt.show()



可以看到在13个特征之中，相关系数大于0.5的特征对：

TAX RAD 0.910228

INDUS NOX 0.763651

NOX AGE 0.731470

TAX INDUS 0.720760

MEDV RM 0.695360

TAX NOX 0.668023

DIS ZN 0.664408

INDUS AGE 0.644779

CRIM RAD 0.625505

NOX RAD 0.611441

INDUS LSTAT 0.603800

LSTAT AGE 0.602339

INDUS RAD 0.595129

NOX LSTAT 0.590879

TAX CRIM 0.582764

LSTAT TAX 0.543993

TAX AGE 0.506456

2.

方法1（最好用的）：接下来我使用PolynomialFeatures进行特征构造，并筛选出相关性较强的结果：

import pandas as pd

import numpy as np

from sklearn.datasets import load\_boston

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

# 读取数据

boston = load\_boston()

data = pd.DataFrame(boston.data, columns=boston.feature\_names)

target = pd.DataFrame(boston.target, columns=['MEDV'])

# 使用PolynomialFeatures进行特征构造

poly = PolynomialFeatures(degree=2, include\_bias=False)

data\_poly = pd.DataFrame(poly.fit\_transform(data), columns=poly.get\_feature\_names(data.columns))

# 合并构造出的特征和原始特征

data\_all = pd.concat([data, data\_poly], axis=1)

# 计算相关性

corr\_matrix = data\_all.corr().abs()

# 找到相关性大于0.9的特征对

high\_corr = np.where(corr\_matrix > 0.9)

high\_corr = [(corr\_matrix.columns[x], corr\_matrix.columns[y]) for x, y in zip(\*high\_corr) if x != y and x < y]

print(high\_corr)

以下是结果：

[('CRIM', 'CRIM'), ('CRIM', 'CRIM INDUS'), ('CRIM', 'CRIM NOX'), ('CRIM', 'CRIM RM'), ('CRIM', 'CRIM AGE'), ('CRIM', 'CRIM DIS'), ('CRIM', 'CRIM RAD'), ('CRIM', 'CRIM TAX'), ('CRIM', 'CRIM PTRATIO'), ('CRIM', 'CRIM LSTAT'), ('ZN', 'ZN'), ('ZN', 'ZN^2'), ('ZN', 'ZN NOX'), ('ZN', 'ZN RM'), ('ZN', 'ZN DIS'), ('ZN', 'ZN TAX'), ('ZN', 'ZN PTRATIO'), ('ZN', 'ZN B'), ('INDUS', 'INDUS'), ('INDUS', 'INDUS^2'), ('INDUS', 'INDUS NOX'), ('INDUS', 'INDUS RM'), ('INDUS', 'INDUS AGE'), ('INDUS', 'INDUS TAX'), ('INDUS', 'INDUS PTRATIO'), ('CHAS', 'CHAS'), ('CHAS', 'INDUS CHAS'), ('CHAS', 'CHAS^2'), ('CHAS', 'CHAS NOX'), ('CHAS', 'CHAS RM'), ('CHAS', 'CHAS AGE'), ('CHAS', 'CHAS DIS'), ('CHAS', 'CHAS TAX'), ('CHAS', 'CHAS PTRATIO'), ('CHAS', 'CHAS B'), ('NOX', 'NOX'), ('NOX', 'NOX^2'), ('NOX', 'NOX AGE'), ('RM', 'RM'), ('RM', 'RM^2'), ('AGE', 'AGE'), ('AGE', 'NOX AGE'), ('AGE', 'RM AGE'), ('AGE', 'AGE^2'), ('AGE', 'AGE PTRATIO'), ('DIS', 'DIS'), ('DIS', 'NOX DIS'), ('DIS', 'RM DIS'), ('DIS', 'DIS^2'), ('DIS', 'DIS PTRATIO'), ('DIS', 'DIS B'), ('RAD', 'TAX'), ('RAD', 'RAD'), ('RAD', 'TAX'), ('RAD', 'INDUS RAD'), ('RAD', 'NOX RAD'), ('RAD', 'RM RAD'), ('RAD', 'AGE RAD'), ('RAD', 'RAD^2'), ('RAD', 'RAD TAX'), ('RAD', 'RAD PTRATIO'), ('RAD', 'TAX^2'), ('RAD', 'TAX PTRATIO'), ('TAX', 'RAD'), ('TAX', 'TAX'), ('TAX', 'INDUS RAD'), ('TAX', 'INDUS TAX'), ('TAX', 'NOX RAD'), ('TAX', 'NOX TAX'), ('TAX', 'RM TAX'), ('TAX', 'AGE RAD'), ('TAX', 'AGE TAX'), ('TAX', 'RAD^2'), ('TAX', 'RAD TAX'), ('TAX', 'RAD PTRATIO'), ('TAX', 'TAX^2'), ('TAX', 'TAX PTRATIO'), ('PTRATIO', 'PTRATIO'), ('PTRATIO', 'PTRATIO^2'), ('B', 'B'), ('B', 'RM B'), ('B', 'PTRATIO B'), ('B', 'B^2'), ('LSTAT', 'LSTAT'), ('LSTAT', 'NOX LSTAT'), ('LSTAT', 'RM LSTAT'), ('LSTAT', 'AGE LSTAT'), ('LSTAT', 'PTRATIO LSTAT'), ('LSTAT', 'LSTAT^2'), ('CRIM', 'CRIM INDUS'), ('CRIM', 'CRIM NOX'), ('CRIM', 'CRIM RM'), ('CRIM', 'CRIM AGE'), ('CRIM', 'CRIM DIS'), ('CRIM', 'CRIM RAD'), ('CRIM', 'CRIM TAX'), ('CRIM', 'CRIM PTRATIO'), ('CRIM', 'CRIM LSTAT'), ('ZN', 'ZN^2'), ('ZN', 'ZN NOX'), ('ZN', 'ZN RM'), ('ZN', 'ZN DIS'), ('ZN', 'ZN TAX'), ('ZN', 'ZN PTRATIO'), ('ZN', 'ZN B'), ('INDUS', 'INDUS^2'), ('INDUS', 'INDUS NOX'), ('INDUS', 'INDUS RM'), ('INDUS', 'INDUS AGE'), ('INDUS', 'INDUS TAX'), ('INDUS', 'INDUS PTRATIO'), ('CHAS', 'INDUS CHAS'), ('CHAS', 'CHAS^2'), ('CHAS', 'CHAS NOX'), ('CHAS', 'CHAS RM'), ('CHAS', 'CHAS AGE'), ('CHAS', 'CHAS DIS'), ('CHAS', 'CHAS TAX'), ('CHAS', 'CHAS PTRATIO'), ('CHAS', 'CHAS B'), ('NOX', 'NOX^2'), ('NOX', 'NOX AGE'), ('RM', 'RM^2'), ('AGE', 'NOX AGE'), ('AGE', 'RM AGE'), ('AGE', 'AGE^2'), ('AGE', 'AGE PTRATIO'), ('DIS', 'NOX DIS'), ('DIS', 'RM DIS'), ('DIS', 'DIS^2'), ('DIS', 'DIS PTRATIO'), ('DIS', 'DIS B'), ('RAD', 'TAX'), ('RAD', 'INDUS RAD'), ('RAD', 'NOX RAD'), ('RAD', 'RM RAD'), ('RAD', 'AGE RAD'), ('RAD', 'RAD^2'), ('RAD', 'RAD TAX'), ('RAD', 'RAD PTRATIO'), ('RAD', 'TAX^2'), ('RAD', 'TAX PTRATIO'), ('TAX', 'INDUS RAD'), ('TAX', 'INDUS TAX'), ('TAX', 'NOX RAD'), ('TAX', 'NOX TAX'), ('TAX', 'RM TAX'), ('TAX', 'AGE RAD'), ('TAX', 'AGE TAX'), ('TAX', 'RAD^2'), ('TAX', 'RAD TAX'), ('TAX', 'RAD PTRATIO'), ('TAX', 'TAX^2'), ('TAX', 'TAX PTRATIO'), ('PTRATIO', 'PTRATIO^2'), ('B', 'RM B'), ('B', 'PTRATIO B'), ('B', 'B^2'), ('LSTAT', 'NOX LSTAT'), ('LSTAT', 'RM LSTAT'), ('LSTAT', 'AGE LSTAT'), ('LSTAT', 'PTRATIO LSTAT'), ('LSTAT', 'LSTAT^2'), ('CRIM INDUS', 'CRIM NOX'), ('CRIM INDUS', 'CRIM RM'), ('CRIM INDUS', 'CRIM AGE'), ('CRIM INDUS', 'CRIM DIS'), ('CRIM INDUS', 'CRIM RAD'), ('CRIM INDUS', 'CRIM TAX'), ('CRIM INDUS', 'CRIM PTRATIO'), ('CRIM INDUS', 'CRIM LSTAT'), ('CRIM CHAS', 'CHAS RAD'), ('CRIM NOX', 'CRIM RM'), ('CRIM NOX', 'CRIM AGE'), ('CRIM NOX', 'CRIM DIS'), ('CRIM NOX', 'CRIM RAD'), ('CRIM NOX', 'CRIM TAX'), ('CRIM NOX', 'CRIM PTRATIO'), ('CRIM NOX', 'CRIM LSTAT'), ('CRIM RM', 'CRIM AGE'), ('CRIM RM', 'CRIM DIS'), ('CRIM RM', 'CRIM RAD'), ('CRIM RM', 'CRIM TAX'), ('CRIM RM', 'CRIM PTRATIO'), ('CRIM RM', 'CRIM LSTAT'), ('CRIM AGE', 'CRIM DIS'), ('CRIM AGE', 'CRIM RAD'), ('CRIM AGE', 'CRIM TAX'), ('CRIM AGE', 'CRIM PTRATIO'), ('CRIM AGE', 'CRIM LSTAT'), ('CRIM DIS', 'CRIM RAD'), ('CRIM DIS', 'CRIM TAX'), ('CRIM DIS', 'CRIM PTRATIO'), ('CRIM DIS', 'CRIM LSTAT'), ('CRIM RAD', 'CRIM TAX'), ('CRIM RAD', 'CRIM PTRATIO'), ('CRIM RAD', 'CRIM LSTAT'), ('CRIM TAX', 'CRIM PTRATIO'), ('CRIM TAX', 'CRIM LSTAT'), ('CRIM PTRATIO', 'CRIM LSTAT'), ('ZN^2', 'ZN NOX'), ('ZN^2', 'ZN RM'), ('ZN^2', 'ZN DIS'), ('ZN^2', 'ZN TAX'), ('ZN^2', 'ZN PTRATIO'), ('ZN^2', 'ZN B'), ('ZN NOX', 'ZN RM'), ('ZN NOX', 'ZN AGE'), ('ZN NOX', 'ZN DIS'), ('ZN NOX', 'ZN TAX'), ('ZN NOX', 'ZN PTRATIO'), ('ZN NOX', 'ZN B'), ('ZN RM', 'ZN DIS'), ('ZN RM', 'ZN TAX'), ('ZN RM', 'ZN PTRATIO'), ('ZN RM', 'ZN B'), ('ZN DIS', 'ZN TAX'), ('ZN DIS', 'ZN PTRATIO'), ('ZN DIS', 'ZN B'), ('ZN TAX', 'ZN PTRATIO'), ('ZN TAX', 'ZN B'), ('ZN PTRATIO', 'ZN B'), ('INDUS^2', 'INDUS NOX'), ('INDUS^2', 'INDUS RM'), ('INDUS^2', 'INDUS AGE'), ('INDUS^2', 'INDUS PTRATIO'), ('INDUS CHAS', 'CHAS^2'), ('INDUS CHAS', 'CHAS NOX'), ('INDUS CHAS', 'CHAS AGE'), ('INDUS CHAS', 'CHAS TAX'), ('INDUS NOX', 'INDUS RM'), ('INDUS NOX', 'INDUS AGE'), ('INDUS NOX', 'INDUS TAX'), ('INDUS NOX', 'INDUS PTRATIO'), ('INDUS RM', 'INDUS AGE'), ('INDUS RM', 'INDUS PTRATIO'), ('INDUS AGE', 'INDUS PTRATIO'), ('INDUS RAD', 'NOX RAD'), ('INDUS RAD', 'NOX TAX'), ('INDUS RAD', 'RM RAD'), ('INDUS RAD', 'AGE RAD'), ('INDUS RAD', 'RAD^2'), ('INDUS RAD', 'RAD TAX'), ('INDUS RAD', 'RAD PTRATIO'), ('INDUS RAD', 'TAX^2'), ('INDUS RAD', 'TAX PTRATIO'), ('INDUS TAX', 'INDUS PTRATIO'), ('INDUS TAX', 'NOX TAX'), ('INDUS TAX', 'AGE TAX'), ('INDUS TAX', 'TAX^2'), ('INDUS TAX', 'TAX PTRATIO'), ('INDUS LSTAT', 'NOX LSTAT'), ('INDUS LSTAT', 'AGE LSTAT'), ('INDUS LSTAT', 'TAX LSTAT'), ('CHAS^2', 'CHAS NOX'), ('CHAS^2', 'CHAS RM'), ('CHAS^2', 'CHAS AGE'), ('CHAS^2', 'CHAS DIS'), ('CHAS^2', 'CHAS TAX'), ('CHAS^2', 'CHAS PTRATIO'), ('CHAS^2', 'CHAS B'), ('CHAS NOX', 'CHAS RM'), ('CHAS NOX', 'CHAS AGE'), ('CHAS NOX', 'CHAS TAX'), ('CHAS NOX', 'CHAS PTRATIO'), ('CHAS NOX', 'CHAS B'), ('CHAS RM', 'CHAS AGE'), ('CHAS RM', 'CHAS DIS'), ('CHAS RM', 'CHAS TAX'), ('CHAS RM', 'CHAS PTRATIO'), ('CHAS RM', 'CHAS B'), ('CHAS AGE', 'CHAS TAX'), ('CHAS AGE', 'CHAS PTRATIO'), ('CHAS AGE', 'CHAS B'), ('CHAS DIS', 'CHAS PTRATIO'), ('CHAS DIS', 'CHAS B'), ('CHAS RAD', 'CHAS TAX'), ('CHAS TAX', 'CHAS PTRATIO'), ('CHAS PTRATIO', 'CHAS B'), ('NOX AGE', 'AGE^2'), ('NOX DIS', 'RM DIS'), ('NOX DIS', 'DIS^2'), ('NOX DIS', 'DIS PTRATIO'), ('NOX DIS', 'DIS B'), ('NOX RAD', 'NOX TAX'), ('NOX RAD', 'RM RAD'), ('NOX RAD', 'AGE RAD'), ('NOX RAD', 'RAD^2'), ('NOX RAD', 'RAD TAX'), ('NOX RAD', 'RAD PTRATIO'), ('NOX RAD', 'TAX^2'), ('NOX RAD', 'TAX PTRATIO'), ('NOX TAX', 'RM TAX'), ('NOX TAX', 'AGE RAD'), ('NOX TAX', 'AGE TAX'), ('NOX TAX', 'RAD TAX'), ('NOX TAX', 'TAX^2'), ('NOX TAX', 'TAX PTRATIO'), ('NOX LSTAT', 'RM LSTAT'), ('NOX LSTAT', 'AGE LSTAT'), ('NOX LSTAT', 'TAX LSTAT'), ('NOX LSTAT', 'PTRATIO LSTAT'), ('NOX LSTAT', 'LSTAT^2'), ('RM AGE', 'AGE^2'), ('RM DIS', 'DIS^2'), ('RM DIS', 'DIS PTRATIO'), ('RM DIS', 'DIS B'), ('RM RAD', 'RM TAX'), ('RM RAD', 'AGE RAD'), ('RM RAD', 'RAD^2'), ('RM RAD', 'RAD TAX'), ('RM RAD', 'RAD PTRATIO'), ('RM RAD', 'TAX^2'), ('RM TAX', 'TAX^2'), ('RM TAX', 'TAX PTRATIO'), ('RM B', 'B^2'), ('RM LSTAT', 'AGE LSTAT'), ('RM LSTAT', 'PTRATIO LSTAT'), ('RM LSTAT', 'LSTAT^2'), ('AGE^2', 'AGE PTRATIO'), ('AGE RAD', 'AGE TAX'), ('AGE RAD', 'RAD^2'), ('AGE RAD', 'RAD TAX'), ('AGE RAD', 'RAD PTRATIO'), ('AGE RAD', 'RAD LSTAT'), ('AGE RAD', 'TAX^2'), ('AGE RAD', 'TAX PTRATIO'), ('AGE TAX', 'TAX^2'), ('AGE LSTAT', 'PTRATIO LSTAT'), ('AGE LSTAT', 'LSTAT^2'), ('DIS^2', 'DIS PTRATIO'), ('DIS^2', 'DIS B'), ('DIS PTRATIO', 'DIS B'), ('RAD^2', 'RAD TAX'), ('RAD^2', 'RAD PTRATIO'), ('RAD^2', 'TAX^2'), ('RAD^2', 'TAX PTRATIO'), ('RAD TAX', 'RAD PTRATIO'), ('RAD TAX', 'TAX^2'), ('RAD TAX', 'TAX PTRATIO'), ('RAD PTRATIO', 'TAX^2'), ('RAD PTRATIO', 'TAX PTRATIO'), ('RAD LSTAT', 'TAX LSTAT'), ('TAX^2', 'TAX PTRATIO'), ('TAX LSTAT', 'PTRATIO LSTAT'), ('PTRATIO LSTAT', 'LSTAT^2')]

方法2：统计值构造法。这个方法可以通过计算原始特征的统计值（例如均值、标准差、最大值、最小值等）来生成新的特征。捕捉到特征之间的非线性关系，提高模型的性能。​下面是代码：

import pandas as pd

import numpy as np

from sklearn.datasets import load\_boston

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import SelectKBest, f\_regression

# 加载数据集

boston = load\_boston()

X = pd.DataFrame(boston.data, columns=boston.feature\_names)

y = pd.DataFrame(boston.target, columns=['MEDV'])

# 构造新特征

X\_new = pd.DataFrame()

# 平均房间数

X\_new['RM'] = X['RM']

X\_new['RM^2'] = X['RM'] \*\* 2

X\_new['RM^3'] = X['RM'] \*\* 3

# 犯罪率

X\_new['CRIM'] = X['CRIM']

X\_new['log(CRIM)'] = np.log(X['CRIM'])

# 市中心距离

X\_new['DIS'] = X['DIS']

X\_new['1/DIS'] = 1 / X['DIS']

# 财产税率

X\_new['B'] = X['B']

X\_new['sqrt(B)'] = np.sqrt(X['B'])

# 选择相关性较强的特征

selector = SelectKBest(score\_func=f\_regression, k=6)

selector.fit(X\_new, y)

mask = selector.get\_support()

# 输出相关性较强的特征

print(X\_new.columns[mask])

结果：



可以看出，房价数量及其平方和立方，犯罪率和其对数，财产税率都和房价有较强的相关性。

方法3：算术运算构造法是指通过对原有特征进行算术运算得到新的特征。我选取了几个比较可能有一定相关性的，可以构造的特征，来构造新的特征。下面是代码：

import pandas as pd

import numpy as np

data = pd.read\_csv('D:/poston/housing.csv', delim\_whitespace=True, header=None)

data.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',

'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

# 构造新特征

data['RM\_LSTAT'] = data['RM'] \* data['LSTAT'] # 房间数乘低收入人群比例

data['RM\_PTRATIO'] = data['RM'] / data['PTRATIO'] # 房间数除以学生与教师比例

data['RM\_TAX'] = data['RM'] / data['TAX'] # 房间数除以房产税

data['AGE\_PTRATIO'] = data['AGE'] / data['PTRATIO'] # 楼龄除以学生与教师比例

data['INDUS\_LSTAT'] = data['INDUS'] \* data['LSTAT'] # 行业比例乘低收入人群比例

# 计算每个新特征与房价的相关系数

corrs = data.corr()['MEDV'].abs().sort\_values(ascending=False)

selected\_features = corrs[corrs >= 0.5].index.tolist()

selected\_features.remove('MEDV') # 移除目标变量

# 打印相关系数较高的特征

print(selected\_features)



出人意料的是“房间数乘低收入人群比例”，“房间数除以学生与教师比例”，“行业比例乘低收入人群比例”都和房价有一定的相关性，我们可以猜测，低收入人群并不能买得起很贵的房子（确信），以及一般来说教育资源越丰富，越大的房子越贵（似乎也是很显然）。

第二题：线性回归算法

这里我直接使用了sklearn之中的线性回归算法

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

# 读取数据

data = pd.read\_csv('D:/poston/housing.csv', delim\_whitespace=True, header=None)

data.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

# 分离特征和标签

X = data.iloc[:, :-1].values

y = data.iloc[:, -1].values

# 划分训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 构建模型

lr = LinearRegression()

lr.fit(X\_train, y\_train)

# 预测测试集

y\_pred = lr.predict(X\_test)

# 计算MSE和RMSE

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

# 打印模型评估指标

print('MSE:', mse)

print('RMSE:', rmse)

print('R^2:', lr.score(X\_test, y\_test))

最终结果如下：

MSE: 24.29111947497339

RMSE: 4.9286021826653235

R^2: 0.6687594935356338

我顺便还测试了一些其他的算法：

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.metrics import r2\_score

# 读取数据

data = pd.read\_csv('D:/poston/housing.csv', header=None, delim\_whitespace=True)

X = data.iloc[:,:-1].values

y = data.iloc[:,-1].values

# 划分训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 构建模型并训练

models = [('Linear Regression', LinearRegression()),

('Decision Tree', DecisionTreeRegressor(random\_state=42)),

('Random Forest', RandomForestRegressor(random\_state=42)),

('Support Vector Regression', SVR(kernel='linear'))]

for name, model in models:

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print(f'{name} R^2: {r2:.3f}')

结果如下：（可以看出随机森林面对这种问题还是有一定优势的）

Linear Regression R^2: 0.669

Decision Tree R^2: 0.858

Random Forest R^2: 0.892

Support Vector Regression R^2: 0.599