线性回归 调用**API**

导入包

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression #线性回归
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_boston
```

数据集分割

X是前13列特征，y是房价

Boston = load_boston()
X = Boston.data
y = Boston.target

分割测试集，训练集，比例为1：4

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=888)
```

```
X_train.shape
```

```
(404, 13)
```

```
y_train.shape
```

```
(404,)
```

回归预测

```
lin_reg = LinearRegression()
```

在训练集拟合

```
lin_reg.fit(X_train,y_train)
```

```
LinearRegression()
```

对测试集评估

```
lin_reg.score(X_test,y_test)
```

```
0.755893222063329
```

对测试集的特征预测得到的房价

```
test_pred = lin_reg.predict(X_test)
test_pred
```

```
array([22.17123289, 35.55703211, 20.8943971 ,
20.19609888, 20.02689174,
       21.21700868, 30.82764123, 28.81457412,
24.62625139, 12.51737207,
       21.66809395, 26.01419263, 20.34518116,
23.2663366 , 22.11437669,
       13.07554361, 17.13768497, 21.97900546,
27.3708199 , 28.01916788,
       24.41448177, 34.36001821, 23.94274892,
26.83103321, 33.1323688 ,
       13.13104618, 20.66162225, 17.3953725 ,
24.90599552, 25.93687134,
       25.43031713, 24.81956864, 17.66949657,
13.13904413, 22.14029635,
       35.50302904, 16.23087515, 19.89717175,
23.06369597, 20.00735812,
```

```
        32.84777096, 25.7275347 , 30.95015644,
23.70226829, 21.41232494,
        13.11049316, 33.15680287, 20.24535073,
-5.21091931, 27.52962052,
        25.0985891 , 13.96531508, 14.09019168,
27.38604613, 14.21771639,
        25.46457847, 17.74201965, 19.33771417,
35.67022047, 26.05415131,
        32.57600176, 24.56533259, 31.75366611,
28.13798769, 31.05895476,
        24.53318847, 23.37137553, 30.55484544,
32.28276335, 20.8739582 ,
        24.73484582, 19.61125447, 36.94909625,
41.49006389, 23.0520405 ,
        18.45466235, 15.93100182, 36.026834  ,
14.71237394,  5.08946136,
        10.76502268, 30.20968135,  2.2021625 ,
26.25917493, 30.30408703,
        22.44745919, -0.433392  , 13.18002787,
33.63212884, 14.98805835,
        17.08333507, 42.66871032, 23.82307954,
22.52783591, 28.80958572,
        20.59345662, 12.73271651, 16.76890465,
26.39741601, 24.49422183,
        24.93537084, 43.81773618])
```

真实的房价

```
y_test
```

```
array([22.4, 32.4, 21.7, 24.5, 16.8, 21.1, 29.4, 28.7,
21.5, 13.6, 21.4,
       24.8, 16.8, 19.4, 21.7, 17.2, 17.1, 18.7, 22.3,
25. , 24.4, 34.6,
       20.1, 22.3, 26.7, 15.6, 19.5, 14.3, 22.7, 21.6,
25. , 24.7, 17.8,
       12.7, 22.7, 46.7, 20.2, 27.1, 25. , 19.9, 32. ,
23.2, 32.2, 19.2,
       21. , 13.4, 31.6, 16.7,  7. , 24.5, 24.2, 11.5,
10.9, 22. , 15.7,
       25.3, 14.9, 15. , 33.4, 28.7, 50. , 25. , 29.9,
26.6, 28.7, 20.5,
       23. , 37. , 30.3, 16.2, 22.2, 19.9, 36. , 48.5,
26.4, 19.8, 17.8,
       38.7, 11.7, 13.8, 15.2, 30.1,  8.1, 30.1, 24. ,
17.8, 13.8,  7.5,
       41.3, 20.1, 13.9, 50. , 20.3, 22.6, 25. , 20.5,
12.8, 19.5, 22. ,
       19.1, 24.6, 50. ])
```

模型评估

计算MSE

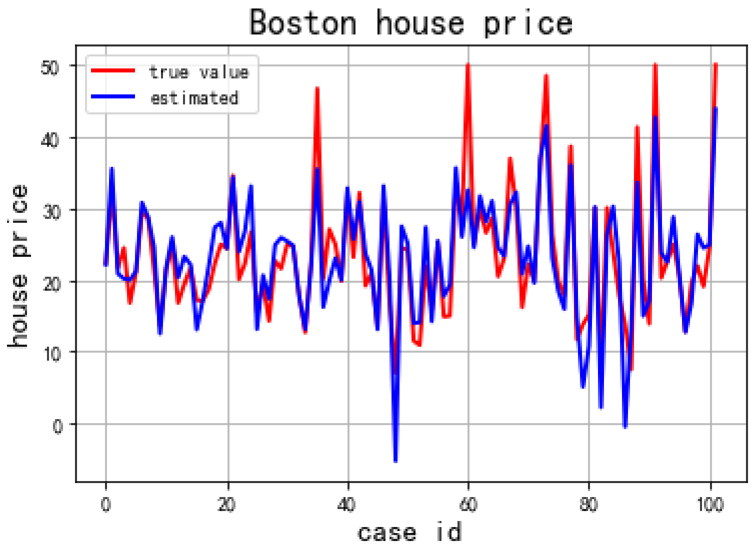
```
# 残差
deviation = lin_reg.predict(X_test) - y_test
```

```
MSE = np.sum(np.sqrt(deviation * deviation))/102
MSE
```

```
3.143244028934462
```

数据可视化

```
import matplotlib as mpl
#对测试集上的标注值与预测值进行可视化呈现
t = np.arange(len(y_test))
mpl.rcParams['font.sans-serif'] = [u'simHei']
mpl.rcParams['axes.unicode_minus'] = False
plt.figure(facecolor='w')
plt.plot(t, y_test, 'r-', lw=2, label=u'true value')
plt.plot(t, test_pred, 'b-', lw=2, label=u'estimated')
plt.legend(loc = 'best')
plt.title(u'Boston house price', fontsize=18)
plt.xlabel(u'case id', fontsize=15)
plt.ylabel(u'house price', fontsize=15)
plt.grid()
plt.show()
```



逻辑回归 调用API

导入所需的包

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split #分割数据集
from sklearn.linear_model import LogisticRegression #逻辑回归模型
from sklearn.preprocessing import StandardScaler #特征标准化
```

数据处理

```
# 1.获取数据集
names = ['Sample code number', 'Clump Thickness',
'Uniformity of Cell Size', 'Uniformity of Cell Shape',
         'Marginal Adhesion', 'Single Epithelial Cell
Size', 'Bare Nuclei', 'Bland Chromatin',
         'Normal Nucleoli', 'Mitoses', 'Class']
data =
pd.read_csv("https://archive.ics.uci.edu/ml/machine-
learning-databases/breast-cancer-wisconsin/breast-
cancer-wisconsin.data",names=names)
# print(data.head(5))
```

```
# 2.缺省值处理
data = data.replace(to_replace="?", value=np.NaN)
data = data.dropna()
```

```
# 3.确顶特征值，目标值
x = data.iloc[:, 1:10] #前:取行数，后:取列数  #从第2列到第
10列 左闭右开
# print(x.head(5))
y = data["Class"] # 取"Class"列作为y
```

```
# 4，分割数据集
# 用train_test_split函数划分出训练集和测试集，测试集占比0.2
x_train, x_test, y_train, y_test = train_test_split(x,
y, test_size=0.2,random_state=44)
# x指数据样本集合，y指样本标签，random_state指随机数种子,用来
保证每次划分出的测试集和数据集是一样的
```

```
# 5，特征标准化
transfer = StandardScaler()
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)
```

logisticRegression

```
# predict（）：训练后返回一个概率值数组，此数组的大小为 n·k，第i行第j列上对应的数值代表模型对此样本属于某类标签的概率值，行和为1。
# 例如预测结果为：[[0.66651809 0.53348191]，代表预测样本的标签是0的概率为0.66651809，1的概率为0.53348191。
lr = LogisticRegression()
lr.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
y_predict = lr.predict(x_test)
y_predict
```

```
array([2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 4, 4, 4, 2, 2, 4,
4, 2, 2, 2, 2, 2,
       2, 4, 4, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 4,
4, 4, 2, 4, 4, 2,
       4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4,
4, 2, 2, 2, 4, 2,
       4, 4, 4, 4, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 4, 4,
2, 4, 4, 2, 2, 2,
       2, 2, 4, 4, 4, 4, 4, 2, 2, 2, 4, 2, 2, 4, 2, 2,
4, 2, 2, 4, 2, 2,
       2, 4, 2, 4, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2,
2, 4, 4, 2, 2, 2,
       2, 2, 2, 2, 2], dtype=int64)
```

性能测评

```
lr.score(x_test,y_test)
```

```
0.9635036496350365
```

线性回归 手写

导入所需要的包

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

读取数据集并进行划分

```
import random
def divideList(f1Name,f2Name,f3Name):
    #n1=random.randint(0,9)
    #n2=random.randint(0,9)
    #while n1==n2:
        #n2=random.randint(0,9)
    n1,n2=8,9
    origin=open(f1Name,'r')
    trainList=open(f2Name,'w')
    testList=open(f3Name,'w')
    totalNum,trainNum,testNum=0,0,0
    line=origin.readline()
    while line:
        totalNum+=1
        if totalNum%10==n1 or totalNum%10==n2:
            testNum+=1
            testList.write(line)
        else:
            trainNum+=1
            trainList.write(line)
        line=origin.readline()
    origin.close()
    trainList.close()
    testList.close()
    print("划分完成")
```

```
    print(totalNum,testNum,trainNum)
    return totalNum,testNum,trainNum

divideList("D:\A_University\Study\机器学习\实验一 线性模型-实验\实验一 线性模型 数据集\housing-data.csv","train_house.list","test_house.list")
```

```
划分完成
506 100 406
```

```
(506, 100, 406)
```

定义预测函数与损失函数

```
#最小二乘,损失函数为平方和损失函数，代价函数为样本的损失函数之和（残差平方和）
#读取训练集建立矩阵
def createMatrix(n:int):
    trainList=open('train_house.list','r')
    line=trainList.readline()
    tempMatrix=[]
    tempMatrixY=[]
    while line:
        #print(line)
        lineList=line.split()
```

```
        #print(lineList)
        numLineList=[]
        numLineList.append(1)
        for i in range(n):
            numLineList.append(float(lineList[i]))
        tempMatrixY.append(float(lineList[n]))
        tempMatrix.append(numLineList[:])
        line=trainList.readline()
    Matrix=np.array(tempMatrix)
    MatrixY=[]
    MatrixY.append(tempMatrixY)
    MatrixY=np.array(MatrixY)
    MatrixY=MatrixY.transpose()
    trainList.close()
    return Matrix,MatrixY
Matrix,MatrixY=createMatrix(13)
#print(Matrix)
#print(MatrixY)
print(Matrix.shape,MatrixY.shape)
print("matrix prepared")
```

```
(406, 14) (406, 1)
matrix prepared
```

存储

```
#求出系数矩阵w并存储
def getModulus(X,Y):
    XT=X.transpose()

 w=np.matmul(np.matmul(np.linalg.inv(np.matmul(XT,X)),
XT),Y)
    np.savetxt("house_module.csv",w,delimiter=',')
    return w
w=getModulus(Matrix,MatrixY)
print(w)
```

```
[[ 3.03260503e+01]
 [-1.06603210e-01]
 [ 5.33266355e-02]
 [ 1.95627446e-02]
 [ 3.12350801e+00]
 [-1.76209205e+01]
 [ 4.23819368e+00]
 [-1.62381880e-03]
 [-1.43234015e+00]
 [ 2.80944461e-01]
 [-1.15857591e-02]
 [-8.26448023e-01]
 [ 1.04721358e-02]
 [-4.98279620e-01]]
```

模型评估

```
#计算结果并算出均方差
def testModulus(w,n,trainNum):
    squareSum=0
    w=w.transpose()
```

```
    testList=open('test_house.list','r')
    line=testList.readline()
    result=[]
    while line:
        lineList=line.split()
        trueY=float(lineList[n])
        x=[]
        tx=[1]
        for i in range(n):
            tx.append(float(lineList[i]))
        x.append(tx)
        x=np.array(x).transpose()
        y=np.matmul(w,x).tolist()[0][0]
        #print(y)
        result.append(y)
        squareSum+=(trueY-y)*(trueY-y)
        line=testList.readline()
        MeanSquareError=squareSum/trainNum
    return result,MeanSquareError
result,MeanSquareError=testModulus(w,13,102)
print("结果")
print(result)
print("均方差")
print(MeanSquareError)
```

结果

```
[19.230639754715316, 11.256835615359188,
17.028868364729643, 15.958932966261179,
14.818448520293858, 19.812088882760452,
22.823165539490805, 22.733499212798357,
17.90603251497334, 9.013741662145339,
33.44433040010851, 21.912893886060694,
21.088181067422205, 17.372059985751353,
23.23054510053417, 21.250681301359606,
25.600830071600186, 30.56475655201793,
36.143443869556556, 35.43938607963793,
20.82791973838014, 22.8207721603875,
23.2848864772496, 19.946124592314423,
15.222799772861878, 19.251376566122968,
19.66230290720322, 14.02811111114303,
7.713063246069102, 8.962871120342346,
32.60234124223507, 28.023652103990596,
22.082383800820747, 25.587522922558104,
28.57094157471051, 31.094662648592525,
33.36216661720469, 32.24514732184633,
32.84765102448311, 34.94199184021854,
17.66527421532308, 23.789120789056913,
27.948732171223146, 24.82921972146062,
32.182842240741586, 35.55807970513463,
32.62452203181156, 28.33847811651492,
19.958078010012798, 21.514638438979766,
43.30485995636287, 35.93588421918691,
40.785519392553546, 38.85578034962575,
35.42777524924723, 30.39825558611735,
27.04324906710765, 27.23930188569258,
19.091523934930247, 29.09950141305045,
32.752943165107105, 28.40531366700132,
18.130718714109406, 24.141953293408793,
19.374797737902437, 20.84754686681005,
```

```
19.182280590490045, 22.098556848798808,
25.92854728755909, 28.024973857595906,
23.0106237905033, 22.36621147848077, 9.23237464777049,
22.672422883632912, 20.40325234316139,
15.918069805834774, 5.356066611719712,
6.04102011453705, 16.03086304352164,
6.441605970712016, 19.32434963453592,
13.407177051272212, 6.316959897855692,
5.609203664069081, 13.484194641854753,
13.90034266714693, 8.442322448797945,
4.688635271425283, 17.979656711248808,
17.390367798073388, 12.217073906788485,
16.96844916501874, 16.746235016808985,
16.793996218938318, 11.151770956817831,
19.041920275476386, 21.009114335808103,
11.89113423715418, 18.8953677812761,
21.14133645645575]
均方差
23.730581522853353
```

逻辑回归 手写

缺省值处理

```
#由于存在缺项，所以进行数据清洗，并删除不需要的编号
def data_wash(fileName:str):
    originFile=open(fileName,'r')
    dataAfterWash=open('dataAfterWash.txt','w')
    line=originFile.readline()
    numData,numLoseData=0,0
```

```
    while line:
        if '?' in line:
            numLoseData+=1
        else:
            index=line.find(',')
            line=line[index+1:]
            dataAfterWash.write(line)
            numData+=1
        line=originFile.readline()
    originFile.close()
    dataAfterWash.close()
    print(numData,numLoseData)
data_wash('breast-cancer-wisconsin.data')
```

```
683 16
```

划分数据集

```
divideList('dataAfterWash.txt','cancerTrain.list','cancerTest.list')
```

```
划分完成
683 136 547
```

```
(683, 136, 547)
```

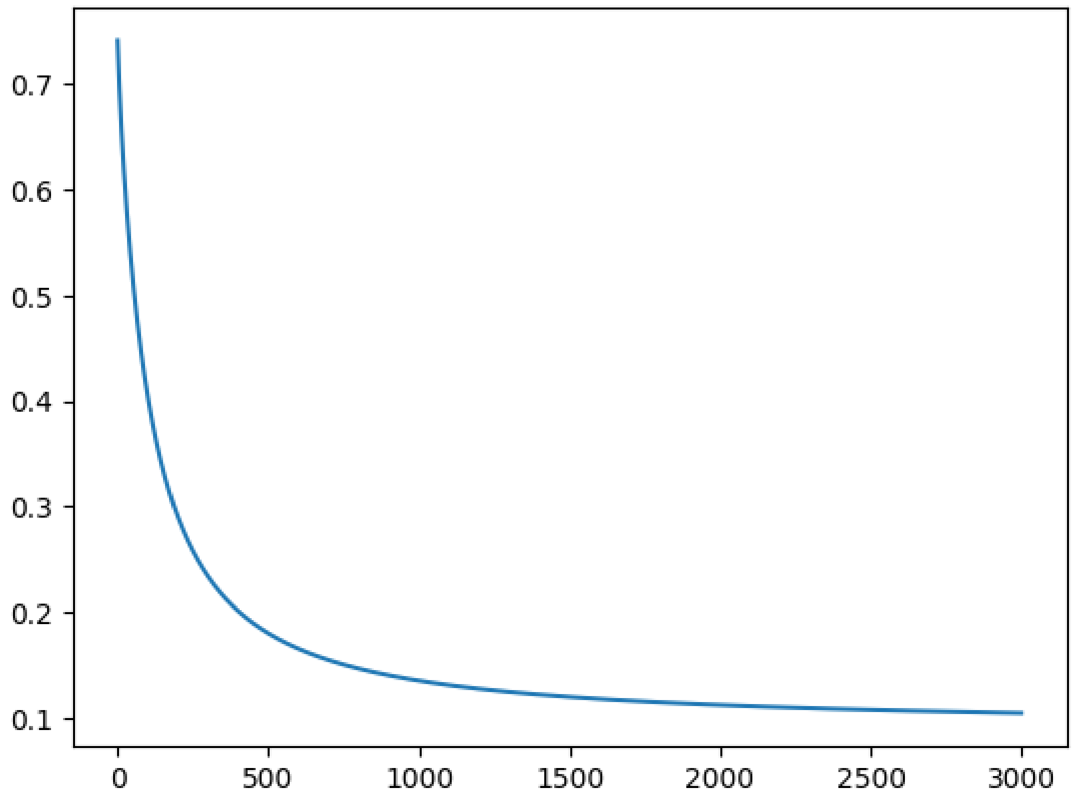
逻辑回归

```
#建立模型，采用批量梯度下降BGD
import time
import math
def
getModule2(learningRate=0.1,trainingRound=3000,file='c
ancerTrain.list',dimension=9,normalization=True)->str:
    time_start=time.time()
    #print("hello")
    #建立列表存储每行向量和结果，设2号种类发生概率为y
    Arrays=[]
    types=[]
    TrainFile=open(file,'r')
    line=TrainFile.readline()

    while line:
        #print(line)
        tempArray1,tempArray2=[],[1]
        Array=line.split(',')
        for i in range(dimension):
            element=float(Array[i])
            if normalization:
                element/=10
            tempArray2.append(element)
        types.append(float(Array[dimension]))
        tempArray1.append(tempArray2)
        #print(np.array(tempArray1).shape)
        Arrays.append(np.array(tempArray1))
        line=TrainFile.readline()
    TrainFile.close()
```

```
    w=[0.1]*(dimension+1)
    Loss=[]
    #损失函数为交叉熵
    for i in range(trainingRound):
        tw=w[:]
        #print(tw)
        #迭代w
        for j in range(dimension+1):
            tempSum=0
            for k in range(len(types)):
                yi=1 if types[k]==2 else 0
                #print(yi)

 #print(np.matmul(np.array([tw]),Arrays[k].T)[0][0])
                hx=1/(1+math.e**(-
np.matmul(np.array([w]),Arrays[k].T)[0][0]))
                #print(hx)
                tempSum+=(hx-yi)*Arrays[k][0][j]
            w[j]-=learningRate*(1/len(types))*tempSum
        #计算每一轮的J(w)
        sumForJw=0
        for k in range(len(types)):
            yi=1 if types[k]==2 else 0
            hx=1/(1+math.e**(-
np.matmul(np.array([w]),Arrays[k].T)[0][0]))
            sumForJw+=yi*math.log(hx)+(1-
yi)*math.log(1-hx)
        Jw=-(1/len(types))*sumForJw
        Loss.append(Jw)
    print(w)
    #画出损失率的折线图
    #print(Loss)
    plt.plot([i for i in range(trainingRound)],Loss)
```

```
    with open("modulusOfQ2.txt",'w') as f:
        f.write(str(w))
    time_end=time.time()
    print("训练时长")
    print(time_end-time_start)
getModule2()
```

```
[5.963701357054275, -1.8713671930762037,
-2.1970458302291203, -2.1805435651659626,
-1.5179412579220435, -0.8237262079624859,
-3.2001096245625456, -1.4115810881402475,
-1.8360851544711918, -0.7396805773402622]
训练时长
102.59360551834106
```



模型评估

```
#对测试集进行检验并判断误差（精度acc，查全率R，查准率p）
def testModulus2(normalization=True,dimension=9):
    TP,FN,FP,TN=0,0,0,0
    with open('modulusOfQ2.txt','r') as f:
        w=eval(f.readline())
    w=np.array([w])
    testFile=open('cancerTest.list','r')
    line=testFile.readline()
    num=0
    while line:
        num+=1
        lineList=line.split(',')
        x=[1]
        for i in range(dimension):
            element=float(lineList[i])
            if normalization:
                element/=10
            x.append(element)
        y=int(lineList[dimension])
        x=np.array([x])
        possible=1/(1+math.e**(-np.matmul(w,x.T)))
        ty=2 if possible>=0.5 else 4
        if ty==2 and y==2:
            TP+=1
        elif y==2 and ty==4:
            FN+=1
        elif y==4 and ty==2:
            FP+=1
        else:
            TN+=1
        line=testFile.readline()
    acc=(TP+TN)/float(num)
```

```
    p=float(TP)/(TP+FP)
    R=float(TP)/(TP+FN)
    print(TP,FN,FP,TN)
    print(acc,p,R)
    testFile.close()
testModulus2()
```

```
93 1 1 41
0.9852941176470589 0.9893617021276596
0.9893617021276596
```

部分特征值预测

为使代码更简洁，结果更清晰，在此只对调用API的版本进行部分特征值预测

逻辑回归 API 选取部分特征

导入所需的包

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split # 分割数据集
from sklearn.linear_model import LogisticRegression # 逻辑回归模型
from sklearn.preprocessing import StandardScaler #特征标准化
from sklearn.feature_selection import VarianceThreshold
```

数据处理

```python
# 1.获取数据集
names = ['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',
         'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
         'Normal Nucleoli', 'Mitoses', 'Class']
data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data",names=names)
```

```python
# 2.缺省值处理
data = data.replace(to_replace="?", value=np.NaN)
data = data.dropna()
```

```
# 3.确顶特征值，目标值
x = data.iloc[:, 1:10] #前:取行数，后:取列数  #从第2列到第10列 左闭右开
y = data["Class"] # 取"Class"列作为y
x.shape
```

```
(683, 9)
```

表示舍弃所有方差小于1的特征

选取了7个特征(全部9个)

```
selector = VarianceThreshold(5)
X_var0 = selector.fit_transform(x)
X_var0.shape
```

```
(683, 7)
```

```
# 4, 分割数据集
# 用train_test_split函数划分出训练集和测试集，测试集占比0.2
x_train, x_test, y_train, y_test =
train_test_split(X_var0, y,
test_size=0.2,random_state=44)
# X指数据样本集合，y指样本标签，random_state指随机数种子,用来
保证每次划分出的测试集和数据集是一样的
```

```
# 5, 特征标准化
transfer = StandardScaler()
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)
```

logisticRegression

```
# predict（）：训练后返回一个概率值数组，此数组的大小为 n·k，
第i行第j列上对应的数值代表模型对此样本属于某类标签的概率值，行和
为1。
# 例如预测结果为：[[0.66651809 0.53348191]，代表预测样本的
标签是0的概率为0.66651809，1的概率为0.53348191。
lr = LogisticRegression()
lr.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
y_predict = lr.predict(x_test)
y_predict
```

```
array([2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 4, 4, 4, 2, 2, 4,
4, 2, 2, 2, 2, 2,
       2, 4, 4, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 4,
4, 4, 2, 4, 4, 2,
       4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4,
4, 2, 2, 2, 4, 2,
       4, 4, 4, 4, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 4, 4,
2, 4, 4, 2, 2, 2,
       4, 2, 4, 4, 4, 4, 4, 2, 2, 2, 4, 2, 2, 4, 2, 2,
4, 2, 2, 4, 2, 2,
       2, 4, 2, 4, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2,
2, 4, 4, 2, 2, 2,
       2, 2, 2, 2, 2], dtype=int64)
```

性能测评

结果0.9562043795620438小于0.9635036496350365（全部特征预测）

```
lr.score(x_test,y_test)
```

```
0.956204379562043‌8
```

总体预测结果比选取全部特征差

线性回归 调用API 选取部分特征

导入包

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression #线性回归
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_boston
from sklearn.feature_selection import VarianceThreshold
```

数据集分割

X是前13列特征，y是房价

```
Boston = load_boston()
X = Boston.data
y = Boston.target
```

表示舍弃所有方差小于1的特征

```
selector = VarianceThreshold(1)
X_var0 = selector.fit_transform(X)
X_var0.shape
```

```
(506, 10)
```

分割测试集，训练集，比例为1：4

```
X_train,X_test,y_train,y_test = 
train_test_split(X_var0,y,test_size=0.2,random_state=8
88)
```

```
X_train.shape
```

```
(404, 10)
```

```
y_train.shape
```

```
(404,)
```

回归预测

```
lin_reg = LinearRegression()
```

在训练集拟合

```
lin_reg.fit(X_train,y_train)
```

```
LinearRegression()
```

对测试集评估

结果0.6703055202990076小于0.755893222063329（全部特征预测）

```
lin_reg.score(X_test,y_test)
```

```
0.6703055202990076
```

对测试集的特征预测得到的房价

```
test_pred = lin_reg.predict(X_test)
test_pred
```

```
array([19.74917836, 33.62289917, 21.47085144,
21.16854662, 21.15438659,
       24.1055678 , 31.85988585, 28.39665121,
22.12418764, 13.2753632 ,
       21.44961075, 26.72640443, 21.15298114,
24.49689913, 19.20866197,
       11.10897137, 16.49685421, 24.90545364,
28.20724796, 28.66021463,
       24.5140849 , 32.0680795 , 24.35164249,
26.27940752, 29.56918298,
       13.16793485, 23.71173681, 17.50317326,
26.1787277 , 27.54628687,
       26.36433963, 26.67358347, 17.62545164,
14.30230693, 22.94235554,
```

```
       32.40809095, 19.57758319, 19.09584697,
22.37117945, 18.5266018 ,
       31.66097631, 27.51909595, 28.93740589,
24.8338167 , 21.98118376,
       10.90025436, 34.68509704, 17.47339194,
-4.02150942, 28.79666777,
       23.63429533, 16.03684064, 15.18412063,
24.99026848, 11.92159478,
       26.63710907, 17.92324348, 17.41920805,
34.33998761, 27.85063967,
       31.38654269, 24.85259985, 31.59429004,
28.02695913, 30.01916596,
       25.91189311, 22.29062086, 29.26823185,
30.07033607, 23.52617169,
       26.18831651, 22.61926004, 37.65708419,
38.75864268, 26.0428238 ,
       19.46422246, 13.88635445, 31.20707281,
15.6629254 ,  4.36045467,
        8.63199608, 30.74383984, -1.80247301,
28.5110415 , 32.34685212,
       18.5772549 ,  0.80295332,  8.79350386,
32.95979623, 12.18235065,
       13.29602901, 39.523532  , 25.83488665,
23.25901934, 29.28031239,
       21.26265185, 13.93175956, 16.20763442,
26.63442387, 25.68667813,
       25.9004037 , 38.45245806])
```

真实的房价

```
y_test
```

```
array([22.4, 32.4, 21.7, 24.5, 16.8, 21.1, 29.4, 28.7,
21.5, 13.6, 21.4,
       24.8, 16.8, 19.4, 21.7, 17.2, 17.1, 18.7, 22.3,
25. , 24.4, 34.6,
       20.1, 22.3, 26.7, 15.6, 19.5, 14.3, 22.7, 21.6,
25. , 24.7, 17.8,
       12.7, 22.7, 46.7, 20.2, 27.1, 25. , 19.9, 32. ,
23.2, 32.2, 19.2,
       21. , 13.4, 31.6, 16.7,  7. , 24.5, 24.2, 11.5,
10.9, 22. , 15.7,
       25.3, 14.9, 15. , 33.4, 28.7, 50. , 25. , 29.9,
26.6, 28.7, 20.5,
       23. , 37. , 30.3, 16.2, 22.2, 19.9, 36. , 48.5,
26.4, 19.8, 17.8,
       38.7, 11.7, 13.8, 15.2, 30.1,  8.1, 30.1, 24. ,
17.8, 13.8,  7.5,
       41.3, 20.1, 13.9, 50. , 20.3, 22.6, 25. , 20.5,
12.8, 19.5, 22. ,
       19.1, 24.6, 50. ])
```

模型评估

计算MSE

结果3.711054178016011大于3.143244028934462（全部特征评估）

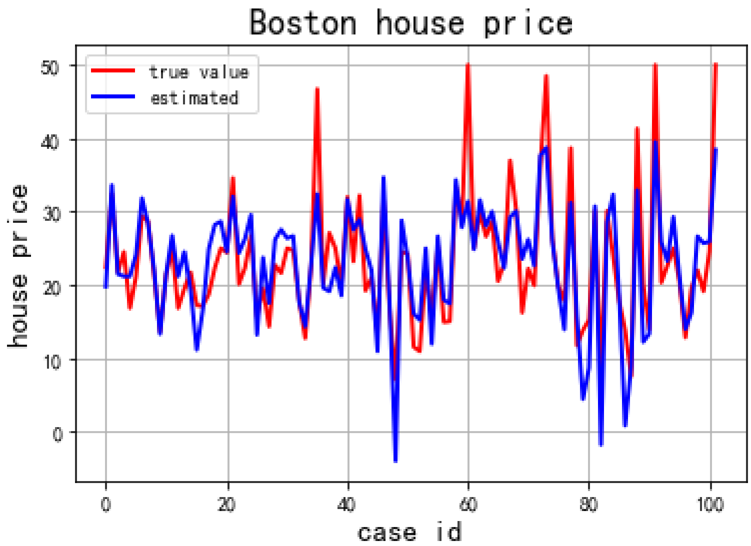
```
# 残差
deviation = lin_reg.predict(X_test) - y_test
```

```
MSE = np.sum(np.sqrt(deviation * deviation))/102
MSE
```

```
3.711054178016011
```

数据可视化

```
import matplotlib as mpl
#对测试集上的标注值与预测值进行可视化呈现
t = np.arange(len(y_test))
mpl.rcParams['font.sans-serif'] = [u'simHei']
mpl.rcParams['axes.unicode_minus'] = False
plt.figure(facecolor='w')
plt.plot(t, y_test, 'r-', lw=2, label=u'true value')
plt.plot(t, test_pred, 'b-', lw=2, label=u'estimated')
plt.legend(loc = 'best')
plt.title(u'Boston house price', fontsize=18)
plt.xlabel(u'case id', fontsize=15)
plt.ylabel(u'house price', fontsize=15)
plt.grid()
plt.show()
```



总体预测结果比选取全部特征差