**特征选择方法总结**

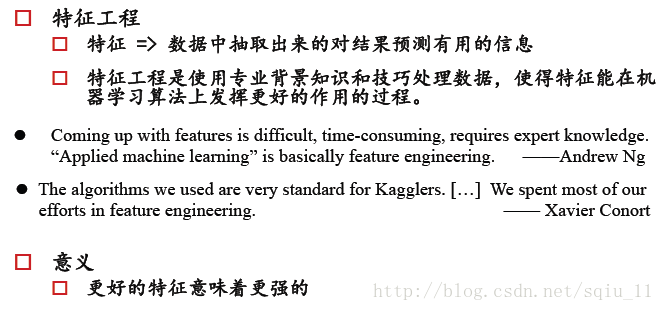
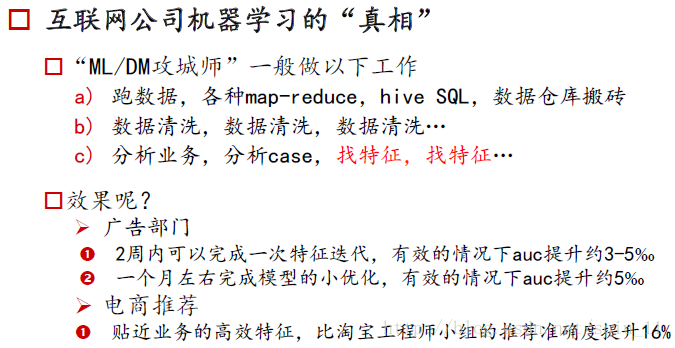
**什么是特征工程？**

定义：特征工程是将原始数据转化为特征，更好表示预测模型处理的实际问题，提升对于未知数据的准确性。它是用目标问题所在的特定领域知识或者自动化的方法来生成、提取、删减或者组合变化得到特征。

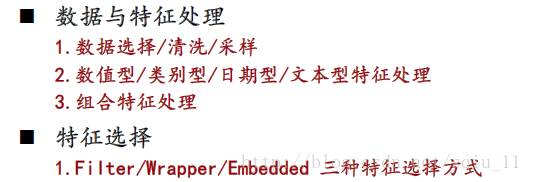
**为什么要特征工程？**

简单的说，你给我的数据能不能直接放到模型里？显然不能，第一，你的数据可能是假(异常值); 第二，你的数据太脏了(各种噪声);第三，你的数据可能不够，或者数据量不平衡(数据采样)；第三，清洗完数据能直接用吗？显然不能！输入模型的数据要和模型输出有关系，没关系的那是噪声！(特征提取或处理)；第四，特征之间属于亲戚关系，你们是一家,不能反映问题！(特征选择)。

**特征工程有何意义？**

这里用[七月在线](https://www.julyedu.com/" \t "_blank)寒老师的PPT做解释：   
  


**如何特征进行工程？**



**本文重点叙述特征选择(Feature selection)**

[sklearn.feature\_selection](http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection)   
有别于降维算法(PCA, SVD)

**方法一：过滤型(Fillter)**

评估单个特征和结果值之间的相关程度，排序留下Top相关的特征部分。

**Person相关系数法：**

import numpy as np

from scipy.stats import pearsonr#引入数值计算包的统计函数(scipy.stats)

np.random.seed(0)#保证每次随机取值相同

size = 300

x = np.random.normal(0, 1, size)#正态分布

print ("Lower noise", pearsonr(x, x + np.random.normal(0, 1, size)))

print ("Higher noise", pearsonr(x, x + np.random.normal(0, 10, size)))

* 1
* 2
* 3
* 4
* 5
* 6
* 7

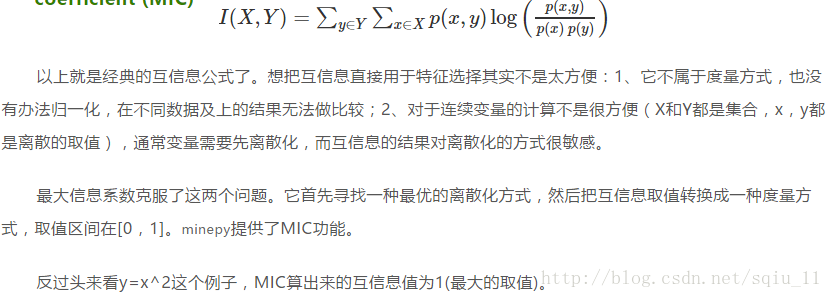
Person相关系数的缺点： 作为特征排序机制，他只对线性关系敏感。如果关系是非线性的，即便两个变量具有一一对应的关系，Pearson相关性也可能会接近0。

x = np.random.uniform(-1, 1, 100000)

print pearsonr(x, x\*\*2)[0]

* 1
* 2

**互信息和最大信息系数 Mutual information and maximal information coefficient (MIC)**



import numpy as np

from minepy import MINE#minepy包——基于最大信息的非参数估计

m = MINE()

x = np.random.uniform(-1, 1, 10000)#均匀分布

m.compute\_score(x, x\*\*2)

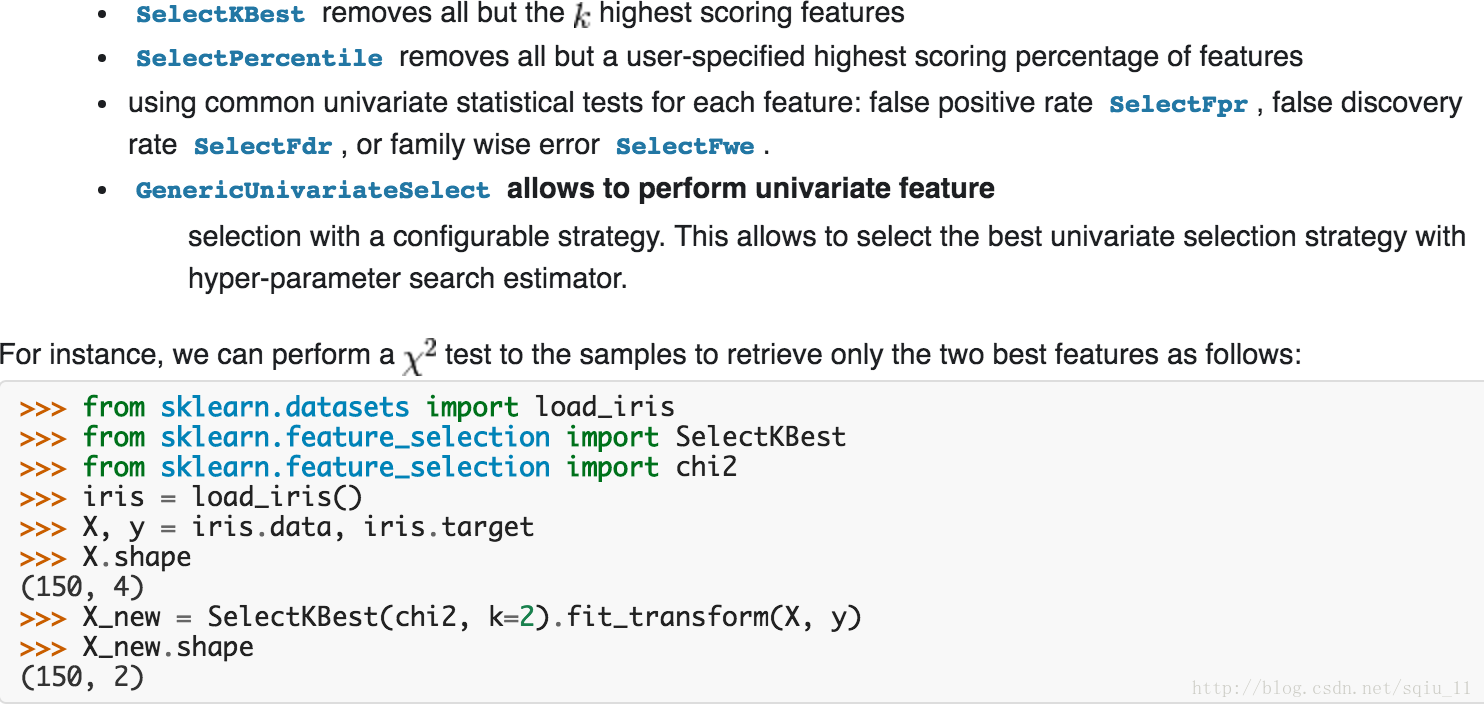
print (m.mic())

**距离相关度**

距离相关系数是为了克服Pearson相关系数的弱点而生的。在x和x^2这个例子中，即便Pearson相关系数是0，我们也不能断定这两个变量是独立的（有可能是非线性相关）；但如果距离相关系数是0，那么我们就可以说这两个变量是独立的。

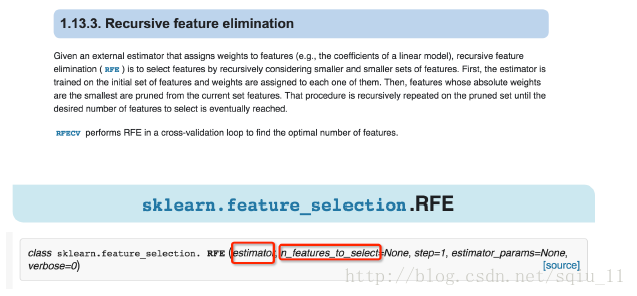
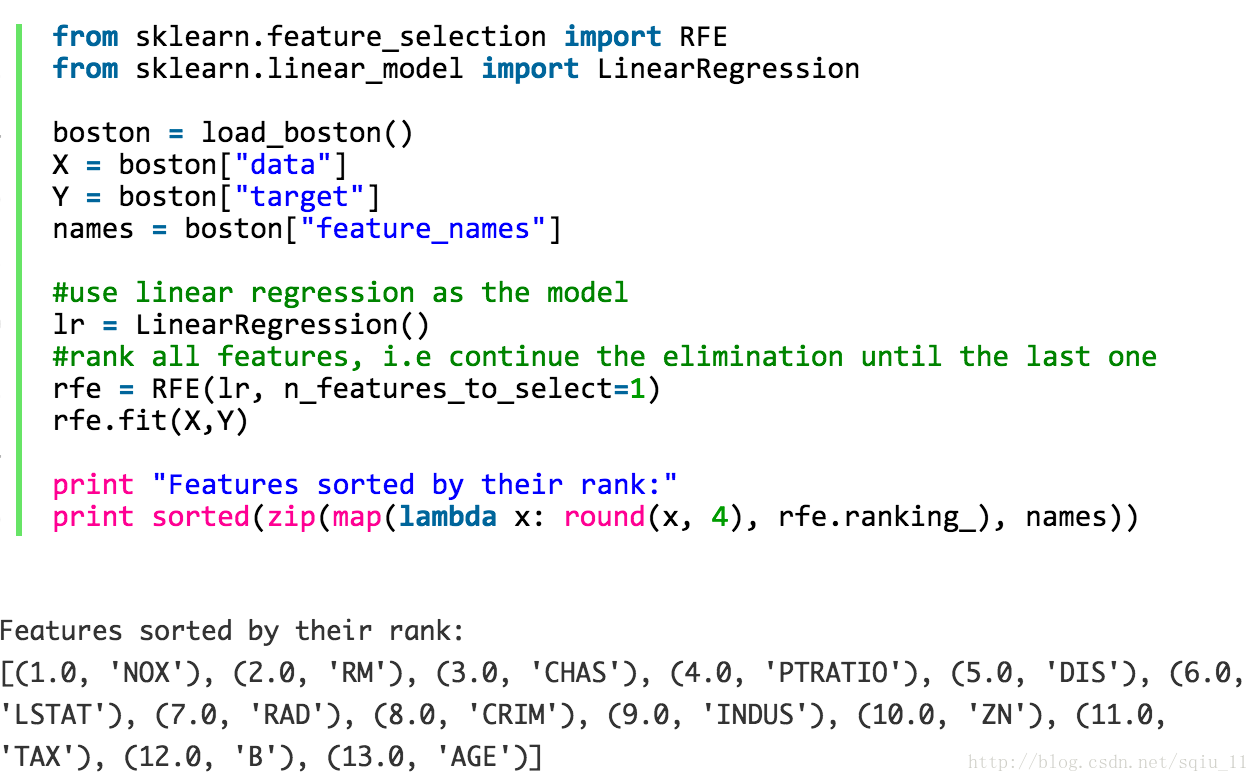
这里写代码片

* 1

过滤型(Fillter)缺点：没有考虑到特征之间的关联作用，可能把有用的关联特征误踢掉。   
[sklearn.feature\_selection](http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection)   


**方法2：包裹型(Wrapper)**

**递归特征删除算法**

把特征选择看做一个特征子集搜索问题，筛选各种特征子集，用模型评估效果。 典型的包裹型算法为 “递归特征删除算法”(recursive feature elimination algorithm)   
比如用逻辑回归，怎么做这个事情呢？   
①用全量特征跑一个模型   
② 根据线性模型的系数(体现相关性)，删掉5-10%的弱特征，观   
察准确率/auc的变化   
③ 逐步进行，直至准确率/auc出现大的下滑停止   
sklearn中用到的是sklearn.feature\_selection.RFE   
  
  
返回的是特征贡献的排序情况

**基于学习模型的特征排序 (Model based ranking)**

这种方法的思路是直接使用你要用的机器学习算法，针对每个单独的特征和响应变量建立预测模型。其实Pearson相关系数等价于线性回归里的标准化回归系数。假如某个特征和响应变量之间的关系是非线性的，可以用基于树的方法（决策树、随机森林）、或者扩展的线性模型等。基于树的方法比较易于使用，因为他们对非线性关系的建模比较好，并且不需要太多的调试。但要注意过拟合问题，因此树的深度最好不要太大，再就是运用交叉验证。

from sklearn.cross\_validation import cross\_val\_score, ShuffleSplit

from sklearn.datasets import load\_boston#波士顿房屋价格预测

from sklearn.ensemble import RandomForestRegressor

#集成学习ensemble库中的随机森林回归RandomForestRegressor

#Load boston housing dataset as an example

boston = load\_boston()

X = boston["data"]

Y = boston["target"]

names = boston["feature\_names"]

rf = RandomForestRegressor(n\_estimators=20, max\_depth=4)

#20个弱分类器，深度为4

scores = []

for i in range(X.shape[1]):#分别让每个特征与响应变量做模型分析并得到误差率

score = cross\_val\_score(rf, X[:, i:i+1], Y, scoring="r2",

cv=ShuffleSplit(len(X), 3, .3))

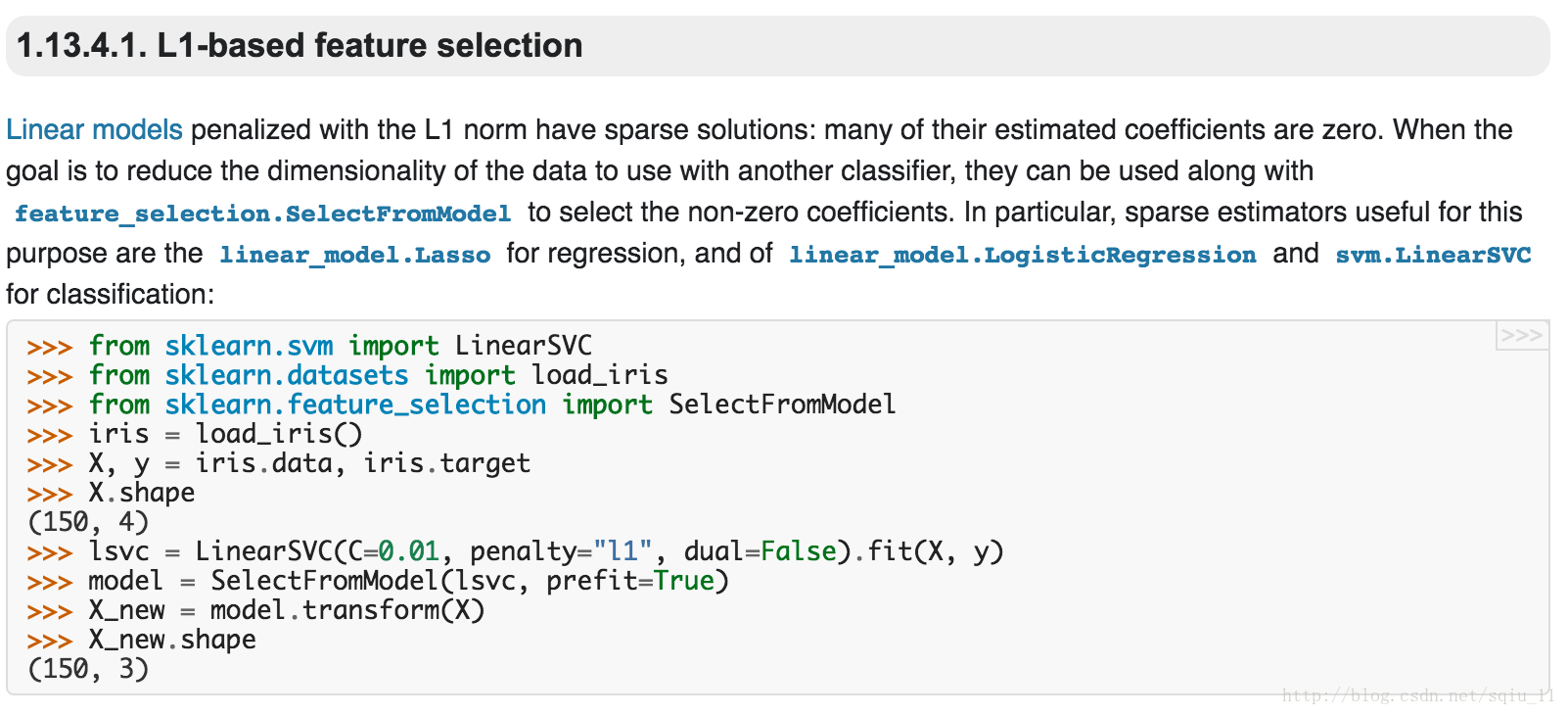
scores.append((round(np.mean(score), 3), names[i]))

print (sorted(scores, reverse=True))#对每个特征的分数排序

**方法3：嵌入型(Embeded)**

根据模型来分析特征的重要性（有别于上面的方式，是从生产的模型权重等）。最常见的方式为用正则化方式来做特征选择。   
举个例子，最早在电商用LR做CTR预估，在3-5亿维的系数特征上用L1正则化的LR模型。剩余2-3千万的feature，意味着其他的feature重要度不够。

**正则化模型**

正则化就是把额外的约束或者惩罚项加到已有模型（损失函数）上，以防止过拟合并提高泛化能力。损失函数由原来的E(X,Y)变为E(X,Y)+alpha||w||，w是模型系数组成的向量（有些地方也叫参数parameter，coefficients），||·||一般是L1或者L2范数，alpha是一个可调的参数，控制着正则化的强度。**当用在线性模型上时，L1正则化和L2正则化也称为Lasso和Ridge。**   
**Lasso**   
L1正则化 是指向量中各个元素绝对值之和，将系数w的l1范数作为惩罚项加到损失函数上，由于正则项非零，这就迫使那些弱的特征所对应的系数变成0。因此L1正则化往往会使学到的模型很稀疏（系数w经常为0），这个特性使得L1正则化成为一种很好的特征选择方法。L1正则化像非正则化线性模型一样也是不稳定的，**如果特征集合中具有相关联的特征，当数据发生细微变化时也有可能导致很大的模型差异。**   
**Ridge**   
L2范数是指向量各元素的平方和然后求平方根，将系数向量的L2范数添加到了损失函数中。由于L2惩罚项中系数是二次方的，这使得L2和L1有着诸多差异，最明显的一点就是，L2正则化会让系数的取值变得平均。对于关联特征，这意味着他们能够获得更相近的对应系数。还是以Y=X1+X2为例，**假设X1和X2具有很强的关联，如果用L1正则化，不论学到的模型是Y=X1+X2还是Y=2X1，惩罚都是一样的，都是2alpha。但是对于L2来说，第一个模型的惩罚项是2alpha，但第二个模型的是4\*alpha。**可以看出，系数之和为常数时，各系数相等时惩罚是最小的，所以才有了L2会让各个系数趋于相同的特点。   
L2正则化对于特征选择来说一种稳定的模型，不像L1正则化那样，系数会因为细微的数据变化而波动。所以L2正则化和L1正则化提供的价值是不同的，L2正则化对于特征理解来说更加有用：表示能力强的特征对应的系数是非零。   


from sklearn.feature\_selection import RFE #包裹型特征选择

from sklearn.preprocessing import StandardScaler #数据标准化

from sklearn.cross\_validation import train\_test\_split #交叉验证

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression #线性回归

from sklearn.datasets import load\_boston

import numpy as np

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import Ridge #L2正则化

from sklearn.linear\_model import Lasso #L1正则化

#数据导入

boston=load\_boston()

scaler = StandardScaler() #数据标准化

X=scaler.fit\_transform(boston.data) #特征变量的数据

y=boston.target #结果-->房价

names=boston.feature\_names #特征名

#算法拟合

lr=LinearRegression() #线性回归算法

rfe=RFE(lr,n\_features\_to\_select=1)

rfe.fit(X,y) #拟合数据

print("原有特征名:")

print("\t",list(names))

print("排序后的特征名:")

print("\t",sorted(zip(map(lambda x: round(x,4),rfe.ranking\_),names))) #对特征进行排序

#提取排序后的属性在原属性列的序列号

rank\_fea=sorted(zip(map(lambda x: round(x,4),rfe.ranking\_),names)) #排序好的特征

rank\_fea\_list=[] #用来装排序的特征的属性名

for i in rank\_fea:

rank\_fea\_list.append(i[1])

index\_list=[0]\*13 #记录特征属性名对应原属性names的序列号

for j,i in enumerate(rank\_fea\_list):

index=list(names).index(i) #获取序列号

index\_list[j]=index

print("排序后特征对应原特征名的序列号：")

print("\t",index\_list)

print("------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------")

'''

#如果想要看一看每个特征与结果之间的散点分布情况的话，请把''' ''''去掉即可，即把注释符号去掉

#给予排序号的每个特征和结果画图，看看每个特征和结果之间的关系

'''

for i in index\_list: #共有13个特征，所以画13张图

plt.figure(names[i]) #每张图以该特征为名

plt.scatter(X[:,i],y) #画出散点图

plt.xlabel(names[i])

plt.ylabel("price house")

#提取排名前n个特征的算法拟合

print("提取排序后的前n个特征向量进行训练:")

for time in range(2,13):

X\_exc=np.zeros((X.shape[0],time)) #把排序好前六个特征向量提取出来,放在X—exc矩阵里

for j,i in enumerate(index\_list[:time]):

X\_exc[:,j]=X[:,i]

X\_train1,X\_test1,y\_train1,y\_test1=train\_test\_split(X\_exc,y)

lr1=LinearRegression()

lr1.fit(X\_train1,y\_train1)

print("\t提取{0}个的特征-->R方值\t".format(time),lr1.score(X\_test1,y\_test1))

print()

#原数据全部特征拟合

print("全部特征向量进行训练：")

X\_train\_raw,X\_test\_raw,y\_train\_raw,y\_test\_raw=train\_test\_split(X,y)

lr\_raw=LinearRegression()

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

print("\t全部特征---->R方值\t",lr\_raw.score(X\_test\_raw,y\_test\_raw))

print()

#只提取一个特征向量

print("只提取一个特征向量进行训练：")

for i in index\_list:

X2=np.zeros((X.shape[0],1))

X2[:,0]=X[:,index\_list[i]]

X\_train2,X\_test2,y\_train2,y\_test2=train\_test\_split(X2,y)

lr2=LinearRegression()

lr2.fit(X\_train2,y\_train2)

print("\t特征",names[i],"---->R方值","\t",lr2.score(X\_test2,y\_test2))

print()

#采取L1正则化的方法

print("采取L1正则化的方法:")

lasso= Lasso(alpha=0.3) #alpha参数由网友用网格搜索方法确定下来的

lasso.fit(X\_train\_raw,y\_train\_raw)

print("\tL1正则化特征---->R方值\t",lasso.score(X\_test\_raw,y\_test\_raw))

print()

#采取L2正则化的方法

print("采取L2正则化的方法")

ridge = Ridge(alpha=10) #alpha参数由网友用网格搜索方法确定下来的

ridge.fit(X\_train\_raw,y\_train\_raw)

print("\tL2正则化特征---->R方值\t",ridge.score(X\_test\_raw,y\_test\_raw))

plt.show() #显示图片

例子中用到基于线性模型用到RFE，正则化等方法，也看出特征选择后对结论的影响。特征选择是进行模型训练前的最后一步，是数据加工过程的最后一步，几乎每个例子中都会涉及到。   
如前面所述，在机器学习领域大部分时间将会与数据打交道，因此，在学习算法的过程中一定要利用python把数据玩儿的很溜，边学python边学数据！！！   
PS   
正则化是一个比较绕人的问题，下一篇会专门讲正则化，会参考几篇CSDN的博客加上自己的理解。