菜菜的scikit-learn课堂09

sklearn中的线性回归大家族

小伙伴们晚上好~o(￣▽￣)ブ

我是菜菜，这里是我的sklearn课堂第九期，今晚的直播内容是线性回归大家族

我的开发环境是**Jupyter lab**，所用的库和版本大家参考：

Python 3.7.1（你的版本至少要3.4以上

Scikit-learn 0.20.1 （你的版本至少要0.20

Numpy 1.15.4, **Pandas** 0.23.4, **Matplotlib** 3.0.2, **SciPy** 1.1.0

请扫码进群领取课件和代码源文件，**扫描二维码后回复"K"就可以进群哦~**

菜菜的scikit-learn课堂09

sklearn中的线性回归大家族

1 概述

1.1 线性回归大家族

回归是一种应用广泛的预测建模技术，这种技术的核心在于预测的结果是连续型变量。决策树，随机森林，支持向量机的分类器等分类算法的预测标签是分类变量，多以{0，1}来表示，而无监督学习算法比如PCA，KMeans的目标根本不是求解出标签，注意加以区别。回归算法源于统计学理论，它可能是机器学习算法中产生最早的算法之一，其在现实中的应用非常广泛，包括使用其他经济指标预测股票市场指数，根据喷射流的特征预测区域内的降水量，根据公司的广告花费预测总销售额，或者根据有机物质中残留的碳-14的量来估计化石的年龄等等，只要一切基于特征预测连续型变量的需求，我们都使用回归技术。

既然线性回归是源于统计分析，我们就可以用不同的角度去理解它。从统计学的角度来看，我们对线性回归有许多要求，比如残差要满足正态分布，要排除共线性等等，然而从机器学习的角度来说，无论服从什么分布，无论是否存在共线性，只要模型效果好就行。我们的课程会从机器学习的角度来为大家讲解回归类算法，如果希望理解统计学角度的小伙伴们，各种统计学教材都可以满足你的需求。

回归需求在现实中非常多，所以我们自然也有各种各样的回归类算法。最著名的就是我们的线性回归和逻辑回归，从他们衍生出了岭回归，Lasso，弹性网，除此之外，还有众多分类算法改进后的回归，比如回归树，随机森林的回归，支持向量回归，贝叶斯回归等等。除此之外，我们还有各种鲁棒的回归：比如RANSAC，Theil-Sen估计，胡贝尔回归等等。考虑到回归问题在现实中的泛用性，回归家族可以说是非常繁荣昌盛，家大业大了。

回归类算法的数学相对简单，相信在经历了逻辑回归，主成分分析与奇异值分解，支持向量机这三个章节之后，大家会觉得线性回归中的数学简直是太容易了。因此，我们在真正应用回归的时候，需要理解的核心，不是要对原理理解得有多么透彻（因为非常简单，所以一看就可以很透彻），而是**我们要明白各个回归算法的优劣，明白如何衡量回归算法的效果，然后了解什么时间该选用什么回归。**

1.2 sklearn中的线性回归

sklearn中的线性模型模块是linear_model，我们曾经在学习逻辑回归的时候提到过这个模块。linear_model包含了多种多样的类和函数，其中逻辑回归相关的类和函数在这里就不给大家列举了。今天的课中我将会为大家来讲解：普通线性回归LinearRegression，岭回归Ridge，LASSO，弹性网。

类/函数	含义
普通线性回归	
linear_model.LinearRegression	使用普通最小二乘法的线性回归
岭回归	
linear_model.Ridge	岭回归，一种将L2作为正则化工具的线性最小二乘回归
linear_model.RidgeCV	带交叉验证的岭回归
linear_model.RidgeClassifier	岭回归的分类器
linear_model.RidgeClassifierCV	带交叉验证的岭回归的分类器
linear_model.ridge_regression	【函数】用正太方程法求解岭回归
LASSO	
linear_model.Lasso	Lasso，使用L1作为正则化工具来训练的线性回归模型
linear_model.LassoCV	带交叉验证和正则化迭代路径的Lasso
linear_model.LassoLars	使用最小角度回归求解的Lasso
linear_model.LassoLarsCV	带交叉验证的使用最小角度回归求解的Lasso
linear_model.LassoLarsIC	使用BIC或AIC进行模型选择的，使用最小角度回归求解的Lasso
linear_model.MultiTaskLasso	使用L1 / L2混合范数作为正则化工具训练的多标签Lasso
linear_model.MultiTaskLassoCV	使用L1 / L2混合范数作为正则化工具训练的，带交叉验证的多标签Lasso
linear_model.lasso_path	【函数】用坐标下降计算Lasso路径
弹性网	
linear_model.ElasticNet	弹性网，一种将L1和L2组合作为正则化工具的线性回归
linear_model.ElasticNetCV	带交叉验证和正则化迭代路径的弹性网
linear_model.MultiTaskElasticNet	多标签弹性网
linear_model.MultiTaskElasticNetCV	带交叉验证的多标签弹性网
linear_model.enet_path	【函数】用坐标下降法计算弹性网的路径

类/函数	含义
最小角度回归	
linear_model.Lars	最小角度回归（Least Angle Regression，LAR）
linear_model.LarsCV	带交叉验证的最小角度回归模型
linear_model.lars_path	【函数】使用LARS算法计算最小角度回归路径或Lasso的路径
正交匹配追踪	
linear_model.OrthogonalMatchingPursuit	正交匹配追踪模型（OMP）
linear_model.OrthogonalMatchingPursuitCV	交叉验证的正交匹配追踪模型（OMP）
linear_model.orthogonal_mp	【函数】正交匹配追踪（OMP）
linear_model.orthogonal_mp_gram	【函数】Gram正交匹配追踪（OMP）
贝叶斯回归	
linear_model.ARDRegression	贝叶斯ARD回归。ARD是自动相关性确定回归（Automatic Relevance Determination Regression），是一种类似于最小二乘的，用来计算参数向量的数学方法。
linear_model.BayesianRidge	贝叶斯岭回归
其他回归	
linear_model.PassiveAggressiveClassifier	被动攻击性分类器
linear_model.PassiveAggressiveRegressor	被动攻击性回归
linear_model.Perceptron	感知机
linear_model.RANSACRegressor	RANSAC（RANdom SAmple Consensus）算法。
linear_model.HuberRegressor	胡博回归，对异常值具有鲁棒性的一种线性回归模型
linear_model.SGDRegressor	通过最小化SGD的正则化损失函数来拟合线性模型
linear_model.TheilSenRegressor	Theil-Sen估计器，一种鲁棒的多元回归模型

2 多元线性回归LinearRegression

2.1 多元线性回归的基本原理

线性回归是机器学习中最简单的回归算法，多元线性回归指的就是一个样本有多个特征的线性回归问题。对于一个有n个特征的样本i而言，它的回归结果可以写作一个几乎人人熟悉的方程：

$$\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_n x_{in}$$

w被统称为模型的参数，其中w_0被称为截距(intercept)，w_1~w_n被称为回归系数(regression coefficient)，有时也是使用θ来表示。这个表达式，其实就和我们小学时就无比熟悉的$y = ax + b$是同样的性质。其中y是我们的目标变量，也就是标签。x_{i1}~x_{in}是样本i上的特征不同特征。如果考虑我们有m个样本，则m个样本的回归结果可以被写作：

$$\hat{\boldsymbol{y}} = w_0 + w_1 \boldsymbol{x_1} + w_2 \boldsymbol{x_2} + \ldots + w_n \boldsymbol{x_n}$$

其中$\boldsymbol{y}$是包含了m个全部的样本的回归结果的列向量。注意，我们通常使用粗体的小写字母来表示列向量，粗体的大写字母表示矩阵。我们可以使用矩阵来表示这个方程，其中$\boldsymbol{w}$可以被看做是一个结构为(1,n)的列矩阵，$\boldsymbol{X}$是一个结构为(m,n)的特征矩阵，则有：

$$\begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \\ \ldots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ x_{11} & x_{12} & x_{13} & \ldots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2n} \\ & & \ldots & & \\ x_{m1} & x_{m2} & x_{m3} & \ldots & x_{mn} \end{bmatrix} * \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_n \end{bmatrix}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}$$

线性回归的任务，就是构造一个预测函数来映射输入的特征矩阵$\boldsymbol{X}$和标签值$\boldsymbol{y}$的线性关系，这个预测函数在不同的教材上写法不同，可能写作$f(x)$，$y_w(x)$，或者$h(x)$等等形式，但无论如何，**这个预测函数的本质就是我们需要构建的模型，而构造预测函数的核心就是找出模型的参数向量$\boldsymbol{w}$。**但我们怎样才能够求解出参数向量呢？

记得在逻辑回归和SVM中，我们都是先定义了损失函数，然后通过最小化损失函数或损失函数的某种变化来将求解参数向量，以此将单纯的求解问题转化为一个最优化问题。在多元线性回归中，我们的损失函数如下定义：

$$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\sum_{i=1}^{m}(y_i - \boldsymbol{X_i}\boldsymbol{w})^2$$

其中y_i是样本i对应的真实标签，$\hat{y}_i$，也就是$\boldsymbol{X_i}\boldsymbol{w}$是样本$i$在一组参数$\boldsymbol{w}$下的预测标签。

首先，这个损失函数代表了向量$y - \hat{y}$的L2范式的平方结果，L2范式的本质是就是欧式距离，即是两个向量上的每个点对应相减后的平方和再开平方，我们现在只实现了向量上每个点对应相减后的平方和，并没有开方，所以我们的损失函数是L2范式，即欧式距离的平方结果。

在这个平方结果下，我们的y和$\hat{y}$分别是我们的真实标签和预测值，也就是说，这个损失函数实在计算我们的真实标签和预测值之间的距离。因此，我们认为这个损失函数衡量了我们构造的模型的预测结果和真实标签的差异，因此我们固然希望我们的预测结果和真实值差异越小越好。所以我们的求解目标就可以转化成：

$$\min_{w} {||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2}^2$$

其中右下角的2表示向量$\boldsymbol{y} - \boldsymbol{Xw}$的L2范式，也就是我们的损失函数所代表的含义。在L2范式上开平方，就是我们的损失函数。这个式子，也正是sklearn当中，用在类*Linear_model.LinerRegression*背后使用的损失函数。我们往往称呼这个式子为SSE（Sum of Sqaured Error，误差平方和）或者RSS（Residual Sum of Squares 残差平方和）。**在sklearn所有官方文档和网页上，我们都称之为RSS残差平方和**，因此在我们的课件中，我们也这样称呼。

2.2 最小二乘法求解多元线性回归的参数

现在问题转换成了求解让RSS最小化的参数向量$\boldsymbol{w}$，**这种通过最小化真实值和预测值之间的RSS来求解参数的方法叫做最小二乘法**。最小二乘法的过程非常简单。求解极值的第一步往往是求解一阶导数并让一阶导数等于0，最小二乘法也不能免俗。因此，我们现在残差平方和RSS上对参数向量$\boldsymbol{w}$求导。这里的过程涉及到少数矩阵求导的内容，需要查表来确定，而要证明这些求导公式则远远地超出了我们课程地要求。在这里，我将矩阵求导的规则给大家列举，如果大家对矩阵求导有更深的兴趣，则可以走维基百科去查看矩阵求导的详细公式的表格：

https://en.wikipedia.org/wiki/Matrix_calculus

接下来，我们就来对$\boldsymbol{w}$求导：

$$\begin{aligned}\frac{\partial RSS}{\partial \boldsymbol{w}} &= \frac{\partial {||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}||_2}^2}{\partial \boldsymbol{w}} \\ &= \frac{\partial (\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})}{\partial \boldsymbol{w}}\end{aligned}$$

$$\because (A-B)^T = A^T - B^T \text{并且} (AB)^T = B^T * A^T$$

$$\begin{aligned}\therefore &= \frac{\partial (\boldsymbol{y}^T - \boldsymbol{w}^T\boldsymbol{X}^T)(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})}{\partial \boldsymbol{w}} \\ &= \frac{\partial (\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w})}{\partial \boldsymbol{w}} \\ &= \frac{\partial \boldsymbol{y}^T\boldsymbol{y} - \partial \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} - \partial \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \partial \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}}{\partial \boldsymbol{w}}\end{aligned}$$

$$\because \text{矩阵求导中，} a\text{为常数，有如下规则：}$$

$$\frac{\partial a}{\partial A} = 0, \quad \frac{\partial A^T B^T C}{\partial A} = B^T C, \quad \frac{\partial C^T B A}{\partial A} = B^T C, \quad \frac{\partial A^T B A}{\partial A} = (B + B^T)A$$

$$\begin{aligned}&= 0 - \boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} \\ &= \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

我们让求导后的一阶导数为0，并且我们的特征矩阵$\boldsymbol{X}$肯定不会是一个所有元素都为0的矩阵，因此我们的$\boldsymbol{X}^T\boldsymbol{X}$不会为0，因此我们可以得到：

$$\begin{aligned}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} &= 0 \\ \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} &= \boldsymbol{X}^T\boldsymbol{y} \\ \text{左乘一个}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\text{则有：} & \\ \boldsymbol{w} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

而此时我们的$\boldsymbol{w}$就是我们参数的最优解。求解出这个参数向量，我们就解出了我们的$\boldsymbol{X}\boldsymbol{w}$，也就能够计算出我们的预测值$\hat{\boldsymbol{y}}$了。对于多元线性回归的理解，到这里就足够了。以算法工程师和数据挖掘工程师为目标的大家，能够手动推导上面的求解过程是基本要求。

2.3 linear_model.LinearRegression

class `sklearn.linear_model.LinearRegression`(*fit_intercept=True*, *normalize=False*, *copy_X=True*, *n_jobs=None*)

参数	含义
fit_intercept	布尔值，可不填，默认为True 是否计算此模型的截距。如果设置为False，则不会计算截距。
normalize	布尔值，可不填，默认为False 当fit_intercept设置为False时，将忽略此参数。如果为True，则特征矩阵X在进入回归之前将会被减去均值（中心化）并除以L2范式（缩放）。如果你希望进行标准化，请在fit数据之前使用preprocessing模块中的标准化专用类StandardScaler。
copy_X	b布尔值，可不填，默认为True 如果为真，将在X.copy()上进行操作，否则的话原本的特征矩阵X可能被线性回归影响并覆盖。
n_jobs	整数或者None，可不填，默认为None 用于计算的作业数。只在多标签的回归和数据量足够大的时候才生效。除非None在joblib.parallel_backend上下文中，否则None统一表示为1。如果输入 -1，则表示使用全部的CPU来进行计算。更多详细内容，请参阅词汇表：https://scikit-learn.org/stable/glossary.html#term-n-jobs

线性回归的类可能是我们目前为止学到的最简单的类，仅有四个参数就可以完成一个完整的算法。并且看得出，这些参数中并没有一个是必填的，更没有对我们的模型有不可替代作用的参数。这说明，线性回归的性能，往往取决于数据本身，而并非是我们的调参能力，线性回归也因此对数据有着很高的要求。幸运的是，现实中大部分连续型变量之间，都存在着或多或少的线性联系。所以线性回归虽然简单，却很强大。顺便一提，sklearn中的线性回归可以处理多标签问题，只需要在fit的时候输入多维度标签就可以了。

- **来做一次回归试试看吧**

1. **导入需要的模块和库**

```
from sklearn.linear_model import LinearRegression as LR
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.datasets import fetch_california_housing as fch #加利福尼亚房屋价值数据集
import pandas as pd
```

2. **导入数据，探索数据**

```
housevalue = fch() #会需要下载，大家可以提前运行试试看

X = pd.DataFrame(housevalue.data) #放入DataFrame中便于查看

y = housevalue.target
```

```
X.shape

y.shape

X.head()

housevalue.feature_names

X.columns = housevalue.feature_names

"""
MedInc：该街区住户的收入中位数
HouseAge：该街区房屋使用年代的中位数
AveRooms：该街区平均的房间数目
AveBedrms：该街区平均的卧室数目
Population：街区人口
AveOccup：平均入住率
Latitude：街区的纬度
Longitude：街区的经度
"""
```

3. **分训练集和测试集**

```
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3,random_state=420)

for i in [Xtrain, Xtest]:
    i.index = range(i.shape[0])

Xtrain.shape
```

4. **建模**

```
reg = LR().fit(Xtrain, Ytrain)
yhat = reg.predict(Xtest)
yhat
```

5. **探索建好的模型**

```
reg.coef_

[*zip(Xtrain.columns,reg.coef_)]

"""
MedInc：该街区住户的收入中位数
HouseAge：该街区房屋使用年代的中位数
AveRooms：该街区平均的房间数目
AveBedrms：该街区平均的卧室数目
Population：街区人口
AveOccup：平均入住率
Latitude：街区的纬度
Longitude：街区的经度
```

```
"""

reg.intercept_
```

属性	含义
coef_	数组，形状为 (n_features,)或者(n_targets, n_features) 线性回归方程中估计出的系数。如果在fit中传递多个标签（当y为二维或以上的时候），则返回的系数是形状为（n_targets，n_features）的二维数组，而如果仅传递一个标签，则返回的系数是长度为n_features的一维数组
intercept_	数组，线性回归中的截距项。

建模的过程在sklearn当中其实非常简单，但模型的效果如何呢？接下来我们来看看多元线性回归的模型评估指标。

2.4 多元线性回归的模型评估指标

回归类算法的模型评估一直都是回归算法中的一个难点，但不像我们曾经讲过的无监督学习算法中的轮廓系数等等评估指标，回归类与分类型算法的模型评估其实是相似的法则——找真实标签和预测值的差异。只不过在分类型算法中，这个差异只有一种角度来评判，那就是是否预测到了正确的分类，而在我们的回归类算法中，我们有两种不同的角度来看待回归的效果：

第一，我们是否预测到了正确的数值。

第二，我们是否拟合到了足够的信息。

这两种角度，分别对应着不同的模型评估指标。

2.4.1 是否预测了正确的数值

回忆一下我们的RSS残差平方和，它的本质是我们的预测值与真实值之间的差异，也就是从第一种角度来评估我们回归的效力，所以RSS既是我们的损失函数，也是我们回归类模型的模型评估指标之一。但是，RSS有着致命的缺点：它是一个无界的和，可以无限地大。我们只知道，我们想要求解最小的RSS，从RSS的公式来看，它不能为负，所以RSS越接近0越好，但我们没有一个概念，究竟多小才算好，多接近0才算好？为了应对这种状况，sklearn中使用RSS的变体，均方误差MSE（mean squared error）来衡量我们的预测值和真实值的差异：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

均方误差，本质是在RSS的基础上除以了样本总量，得到了每个样本量上的平均误差。有了平均误差，我们就可以将平均误差和我们的标签的取值范围在一起比较，以此获得一个较为可靠的评估依据。在sklearn当中，我们有两种方式调用这个评估指标，一种是使用sklearn专用的模型评估模块metrics里的类mean_squared_error，另一种是调用交叉验证的类cross_val_score并使用里面的scoring参数来设置使用均方误差。

```
from sklearn.metrics import mean_squared_error as MSE
MSE(yhat,Ytest)

y.max()
y.min()

cross_val_score(reg,X,y,cv=10,scoring="mean_squared_error")

#为什么报错了？来试试看！
import sklearn
sorted(sklearn.metrics.SCORERS.keys())

cross_val_score(reg,X,y,cv=10,scoring="neg_mean_squared_error")
```

欢迎来的线性回归的大坑一号：均方误差为负。

我们在决策树和随机森林中都提到过，虽然均方误差永远为正，但是sklearn中的参数scoring下，均方误差作为评判标准时，却是计算"负均方误差"（neg_mean_squared_error）。这是因为sklearn在计算模型评估指标的时候，会考虑指标本身的性质，均方误差本身是一种误差，所以被sklearn划分为模型的一种损失(loss)。在sklearn当中，所有的损失都使用负数表示，因此均方误差也被显示为负数了。真正的均方误差MSE的数值，其实就是neg_mean_squared_error去掉负号的数字。

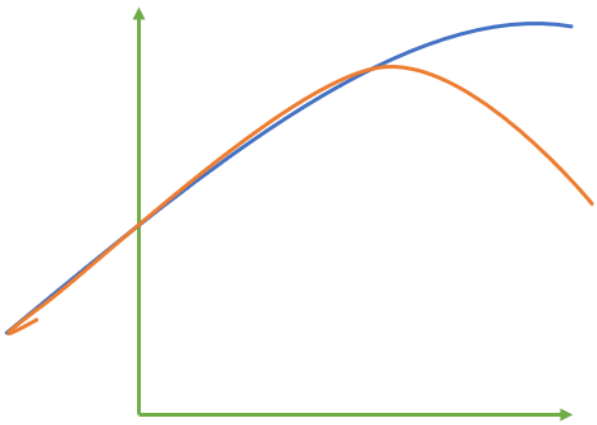
除了MSE，我们还有与MSE类似的MAE（Mean absolute error，绝对均值误差）：

$$MAE = \frac{1}{m}\sum_{i=0}^{m-1}|y_i - \hat{y}_i|$$

其表达的概念与均方误差完全一致，不过在真实标签和预测值之间的差异外我们使用的是L1范式（绝对值）。**现实使用中，MSE和MAE选一个来使用就好了。**在sklearn当中，我们使用命令from sklearn.metrics import mean_absolute_error来调用MAE，同时，我们也可以使用交叉验证中的scoring = "neg_mean_absolute_error"，以此在交叉验证时调用MAE。

2.4.2 是否拟合了足够的信息

对于回归类算法而言，只探索数据预测是否准确是不足够的。除了数据本身的数值大小之外，我们还希望我们的模型能够捕捉到数据的"规律"，比如数据的分布规律，单调性等等，而是否捕获了这些信息并无法使用MSE来衡量。



来看这张图，其中红色线是我们的真实标签，而蓝色线是我们的拟合模型。这是一种比较极端，但的确可能发生的情况。这张图像上，前半部分的拟合非常成功，看上去我们的真实标签和我们的预测结果几乎重合，但后半部分的拟合却非常糟糕，模型向着与真实标签完全相反的方向去了。对于这样的一个拟合模型，如果我们使用MSE来对它进行判断，它的MSE会很小，因为大部分样本其实都被完美拟合了，少数样本的真实值和预测值的巨大差异在被均分到每个样本上之后，MSE就会很小。但这样的拟合结果必然不是一个好结果，因为一旦我的新样本是处于拟合曲线的后半段的，我的预测结果必然会有巨大的偏差，而这不是我们希望看到的。所以，我们希望找到新的指标，除了判断预测的数值是否正确之外，还能够判断我们的模型是否拟合了足够多的，数值之外的信息。

在我们学习降维算法PCA的时候，我们提到我们使用方差来衡量数据上的信息量。如果方差越大，代表数据上的信息量越多，而这个信息量不仅包括了数值的大小，还包括了我们希望模型捕捉的那些规律。为了衡量模型对数据上的信息量的捕捉，我们定义了R^2和**可解释性方差分数(explained_variance_score，EVS)**来帮助我们：

$$R^2 = 1 - \frac{\sum_{i=0}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{m}(y_i - \bar{y})^2} = 1 - \frac{RSS}{\sum_{i=0}^{m}(y_i - \bar{y})^2}$$

$$EVS = 1 - \frac{Var(y_i - \hat{y})}{Var(y_i)}$$

其中$\boldsymbol{y}$是我们的真实标签，$\hat{\boldsymbol{y}}$是我们的预测结果，$\bar{\boldsymbol{y}}$是我们的均值，Var表示方差。方差的本质是任意一个y值和样本均值的差异，差异越大，这些值所带的信息越多。在R^2和EVS中，分子是真实值和预测值之差的差值，也就是我们的模型没有捕获到的信息总量，分母是真实标签所带的信息量，所以两者都**衡量 1 - 我们的模型没有捕获到的信息量占真实标签中所带的信息量的比例**，所以，两者都是越接近1越好。

R^2我们也可以使用三种方式来调用，一种是直接从metrics中导入r2_score，输入预测值和真实值后打分。第二种是直接从线性回归LinearRegression的接口score来进行调用。第三种是在交叉验证中，输入"r2"来调用。EVS有两种调用方法，可以从metrics中导入，也可以在交叉验证中输入”explained_variance“来调用。

```
#调用R2
from sklearn.metrics import r2_score
r2_score(yhat,Ytest)

r2 = reg.score(Xtest,Ytest)
r2
```

我们现在踩到了线性回归的大坑二号：相同的评估指标不同的结果。

为什么结果会不一致呢？这就是回归和分类算法的不同带来的坑。

在我们的分类模型的评价指标当中，我们进行的是一种 if a == b的对比，这种判断和if b == a其实完全是一种概念，所以我们在进行模型评估的时候，从未踩到我们现在在的这个坑里。然而看R2的计算公式，R2明显和分类模型的指标中的accuracy或者precision不一样，R2涉及到的计算中对预测值和真实值有极大的区别，必须是预测值在分子，真实值在分母，所以我们在调用metrcis模块中的模型评估指标的时候，必须要检查清楚，指标的参数中，究竟是要求我们先输入真实值还是先输入预测值。

```
#使用shift tab键来检查究竟哪个值先进行输入
r2_score(Ytest,yhat)

#或者你也可以指定参数，就不必在意顺序了
r2_score(y_true = Ytest,y_pred = yhat)

cross_val_score(reg,X,y,cv=10,scoring="r2").mean()

#调用EVS
from sklearn.metrics import explained_variance_score as EVS
EVS(Ytest,yhat)

cross_val_score(reg,X,y,cv=10,scoring="explained_variance")
```

我们观察到，虽然我们在加利福尼亚房子价值数据集上的MSE相当小，但我们的R^2却不高，这证明我们的模型比较好地拟合了数据的数值，却没有能正确拟合数据的分布。让我们与绘图来看看，究竟是不是这样一回事。我们可以绘制一张图上的两条曲线，一条曲线是我们的真实标签Ytest，另一条曲线是我们的预测结果yhat，两条曲线的交叠越多，我们的模型拟合就越好。

```
import matplotlib.pyplot as plt
sorted(Ytest)

plt.plot(range(len(Ytest)),sorted(Ytest),c="black",label= "Data")
plt.plot(range(len(yhat)),sorted(yhat),c="red",label = "Predict")
plt.legend()
plt.show()
```

可见，虽然我们的大部分数据被拟合得比较好，但是图像的开头和结尾处却又着较大的拟合误差。如果我们在图像右侧分布着更多的数据，我们的模型就会越来越偏离我们真正的标签。这种结果类似于我们前面提到的，虽然在有限的数据集上将数值预测正确了，但却没有正确拟合数据的分布，如果有更多的数据进入我们的模型，那数据标签被预测错误的可能性是非常大的。

思考

细心的小伙伴可能已经注意到了，其实EVS和R^2是异曲同工的，两者都是衡量 1 - 没有捕获到的信息占总信息的比例，EVS和R^2难道不应该相等吗？但从我们的结果来看，两者虽然相似，但却并不完全相等，这中间的差值究竟是什么呢？R^2和**EVS**有什么不同？

现在，来看一组有趣的情况：

```
import numpy as np
rng = np.random.RandomState(42)
X = rng.randn(100, 80)
y = rng.randn(100)
cross_val_score(LR(), X, y, cv=5, scoring='r2')
```

好了，现在我们跋山涉水来到了线性回归的三号大坑：负的R^2。

许多学习过统计理论的小伙伴此时可能会感觉到，太坑了！sklearn真的是设置了太多障碍，均方误差是负的，R^2也是负的，不能忍了。还有的小伙伴可能觉得，这个R^2名字里都带平方了，居然是负的！无论如何，我们再来看看R^2的计算公式：

$$R^2 = 1 - \frac{\sum_{i=0}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{m}(y_i - \bar{y})^2} = 1 - \frac{RSS}{\sum_{i=0}^{m}(y_i - \bar{y})^2}$$

第一次学习机器学习的小伙伴，可能会感觉没什么问题了，R^2是1减一个数，后面的部分只要大于1的话R^2完全可以小于0。但是学过机器学习，尤其是在统计学上有基础的小伙伴可能会坐不住了：这不对啊！

一直以来，众多的机器学习教材中都有这样的解读：

除了RSS之外，我们还有解释平方和ESS（Explained Sum of Squares，也叫做SSR回归平方和）以及总离差平方和TSS（Total Sum of Squares，也叫做SST总离差平方和）。解释平方和ESS定义了我们的预测值和样本均值之间的差异，而总离差平方和定义了真实值和样本均值之间的差异，两个指标分别写作：

$$TSS = \sum_{i=0}^{m}(y_i - \bar{y})^2$$
$$ESS = \sum_{i=0}^{m}(\hat{y}_i - \bar{y})^2$$

而我们有公式：

$$TSS = RSS + ESS \tag{1}$$

看我们的R^2的公式，如果带入我们的TSS和ESS，那就有：

$$R^2 = 1 - \frac{RSS}{TSS} = \frac{TSS - RSS}{TSS} = \frac{ESS}{TSS}$$

而ESS和TSS都带平方，所以必然都是正数，那R^2怎么可能是负的呢？

好了，颠覆认知的时刻到来了——**公式TSS = RSS + ESS不是永远成立的！**就算所有的教材和许多博客里都理所当然这样写了大家也请抱着怀疑精神研究一下，你很快就会发现很多新世界。我们来看一看我们是如何证明(1)这个公式的：

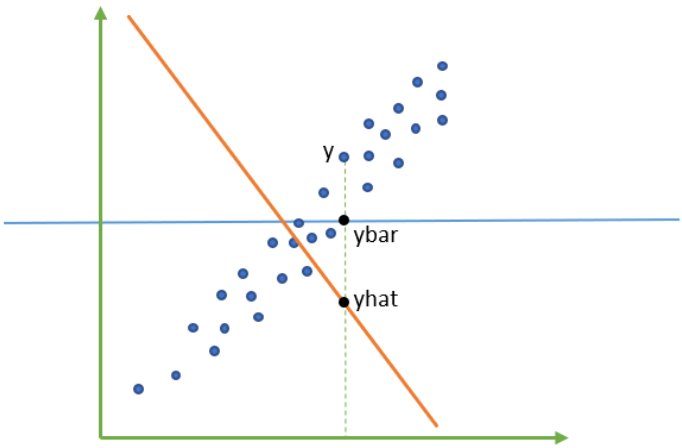
$$\begin{aligned} TSS &= \sum_{i=0}^{m}(y_i - \bar{y})^2 \\ &= \sum_{i=0}^{m}(y_i - \hat{y}_i + \hat{y}_i - \bar{y})^2 \\ &= \sum_{i=0}^{m}(y_i - \hat{y}_i)^2 + \sum_{i=0}^{m}(\hat{y}_i - \bar{y})^2 + 2\sum_{i=0}^{m}(y_i - \hat{y}_i)(\hat{y}_i - \bar{y}) \\ &= RSS + ESS + 2\sum_{i=0}^{m}(y_i - \hat{y}_i)(\hat{y}_i - \bar{y}) \end{aligned}$$

两边同时除以TSS，则有：

$$R^2 = 1 - \frac{RSS}{TSS} = \frac{ESS + 2\sum_{i=0}^{m}(y_i - \hat{y}_i)(\hat{y}_i - \bar{y})}{TSS}$$

许多教材和博客中让$2\sum_{i=0}^{m}(y_i - \hat{y}_i)(\hat{y}_i - \bar{y})$这个式子为0，公式$TSS = RSS + ESS$自然就成立了，但要让这个式子成立是有条件的。现在有了这个式子的存在，R^2就可以是一个负数了。只要我们的$(y_i - \hat{y}_i)$衡量的是真实值到预测值的距离，而$(\hat{y}_i - \bar{y})$衡量的是预测值到均值的距离，只要当这两个部分的符号不同的时候，我们的式子(2)就为负，而R^2就有机会是一个负数。

看下面这张图，蓝色的横线是我们的均值线$\bar{y}$，橙色的线是我们的模型$\hat{y}$，蓝色的点是我们的样本点。现在对于x_i来说，我们的真实标签减预测值的值$(y_i - \hat{y}_i)$为正，但我们的预测值$(\hat{y}_i - \bar{y})$却是一个负数，这说明，数据本身的均值，比我们对数据的拟合模型本身更接近数据的真实值，那我们的模型就是废的，完全没有作用，类似于分类模型中的分类准确率为50%，不如瞎猜。



也就是说，当我们的R^2显示为负的时候，这证明我们的模型对我们的数据的拟合非常糟糕，模型完全不能使用。所有，一个负的R^2是合理的。当然了，现实应用中，如果你发现你的线性回归模型出现了负的R^2，不代表你就要接受他了，首先检查你的建模过程和数据处理过程是否正确，也许你已经伤害了数据本身，也许你的建模过程是存在bug的。如果你检查了所有的代码，也确定了你的预处理没有问题，但你的R^2也还是负的，那这就证明，线性回归模型不适合你的数据，试试看其他的算法吧。

【完整版】2.5 多元线性回归在多标签回归上的推广

【完整版】2.6 案例：分箱在多元线性回归中的重要作用

【完整版】3 多元非线性回归PolynomialFeatures

【完整版】3.1 线性回归使用核函数

【完整版】3.2 案例：多元线性回归在非线性情况的推广

【完整版】4 岭回归Ridge

【完整版】4.1 岭回归的基本原理

【完整版】4.2 岭回归的最佳参数选择

【完整版】4.3 多标签岭回归

【完整版】5 Lasso

【完整版】5.1 Lasso的基本原理

【完整版】5.2 Lasso的最佳参数选择

【完整版】5.3 多标签Lasso

【完整版】6 弹性网ElasticNet

【完整版】6.1 弹性网的基本原理

【完整版】6.2 弹性网的最佳参数选择

【完整版】6.3 多标签的弹性网

【完整版】7 附录

【完整版】7.1 线性回归LinearRegression的参数，属性和接口列表

【完整版】7.2 非线性回归PolynomialFeatures的参数，属性和接口列表

【完整版】7.3 岭回归Ridge的参数，属性和接口列表

【完整版】7.4 Lasso的参数，属性和接口列表

【完整版】7.5 弹性网ElasticNet的参数，属性和接口列表