使用scikit-learn构建模型

加载datasets模块中数据集

datasets模块常用数据集加载函数及其解释

- sklearn库的datasets模块集成了部分数据分析的经典数据集，可以使用这些数据集进行数据预处理、建模等操作，熟悉sklearn的数据处理流程和建模流程。
- 使用sklearn进行预处理会用到sklearn提供的统一接口—转换器。
- 加载后的数据集可以视为一个字典，几乎所有的sklearn数据集均可以使用data，target，feature_names, DESCR分别获取数据集的数据、标签、特征名称和描述信息。
- datasets模块常用数据集的加载函数与解释如下表所示。

数据集加载函数	数据集任务类型	数据集加载函数	数据集任务类型
load_boston	回归	load_breast_cancer	分类，聚类
fetch_California_housing	回归	load_iris	分类，聚类
load_digits	分类	load_wine	分类

In [176]:
```
from sklearn.datasets import load_boston
```

In [177]:
```
data = load_boston()
```

In [178]:
```
data.keys()
```

Out[178]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [180]:
```
data['data']
```

Out[180]:
```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]])
```

将数据集划分为训练集和测试集

常用划分方法:

在数据分析过程中，为了保证模型在实际系统中能够起到预期作用，一般需要将样本分为独立的三部分:

- **训练集**：用于估计模型。
- **验证集**：用于确定网络结构或者控制模型复杂程度的参数。
- **测试集**：用于检验最优的模型性能。

典型的划分方式是训练集占总样本的50%，验证集和测试集各占25%。

K-fold cross validation

当数据总量较少的时候，使用上面的方法将数据划分为三部分就不合适了。

常用的方法：

- 留少部分做测试集，
- 对其余N个样本采用K-fold cross validation。

将数据集划分为训练集和测试集

train_test_split函数

- sklearn的model_selection模块提供了train_test_split函数，能够对数据集进行拆分，其使用格式如下：

sklearn.model_selection.train_test_split(*arrays,**options)

参数名称	说明
*arrays	接收一个或多个数据集。代表需要划分的数据集，若为分类回归则分别传入数据和标签，若为聚类则传入数据。无默认。
test_size	接收float，int，None类型的数据。代表测试集的大小。如果传入的为float类型的数据则需要限定在0-1之间，代表测试集在总体中的占比；如果传入的为int类型的数据，则表示测试集记录的绝对数目。改参数与train_size可以只传入一个。
train_size	接收float，int，None类型的数据。代表训练集的大小。
random_state	接收int。代表随机种子的编号，相同(不同)随机种子的编号产生相同(不同)的结果。默认None。
shuffle	接收boolean。代表是否进行有放回抽样。若该参数取值为True则stratify参数必须不能空。
sratify	接收array或者none。如果不是none，则使用传入的标签进行分层抽样。

In [182]:
```
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(data['data'], data['target'], test_size=0.2)
```

使用sklearn转换器进行数据预处理与降维

sklearn转换器三个方法

sklearn把相关的功能封装为转换器（transformer）。使用sklearn转换器能够实现对传入的NumPy数组进行标准化处理，归一化处理，二值化处理，PCA降维等操作。

方法名称	说明
fit	fit方法主要通过分析特征和目标值，提取有价值的信息，这些信息可以是统计量，也可以是权值系数。
transform	transform方法主要用来对特征进行转换。从可利用信息的角度可分为无信息转换和有信息转换。无信息转换是指不利用其他信息转换，比如指数和对数函数转换等。有信息转换根据是否利用目标值向量又可以分为无监督转换和有监督转换。无监督转换指只利用特征的统计信息的转换，比如标准化和PCA降维等。有监督转换既利用了特征信息又利用了目标值信息的转换，比如通过模型选择特征和LDA降维等。
fit_transform	fit_transform方法就是先调用fit方法，再调用transform方法。

sklearn部分预处理函数与其作用

函数名称	说明
MinmaxScaler	对特征进行离差标准化
StandardScaler	对特征进行标准差标准化
Normalizer	对特征进行归一化
Binarizer	对定量特征进行二值化处理
OneHotEncoder	对定量特征进行独热编码处理
FunctionTransformer	对特征进行自定义函数变换

In [183]:
```
from sklearn.preprocessing import MinMaxScaler
model = MinMaxScaler().fit(train_data)
train_data_t = model.transform(train_data)
test_data_t = model.transform(test_data)
```

PCA降维算法函数常用参数及其作用

函数名称	说明
n_components	原始数据降低到的维度。未指定时，代表所有特征均被保留下来。
copy	接收bool。代表是否在运行算法时将原始数据复制一份，如果为True，则运行后，原始数据的值不会有任何改变；如果为False，则运行PCA算法后，原始训练数据的值会发生改变。默认True。
whiten	Boolean。如果为True，对降维后的数据的每个特征进行归一化，让方差都为1. 默认False。
svd_solver	接收tring{‘auto’，‘full’，‘arpack’，‘randomized’}。代表使用的svd算法。‘randomized’一般适用于数据量大，数据维度多，同时主要数目又低的PCA降维，它使用了一些加快SVD的随机算法。‘full’是使用SciPy库实现的传统SVD算法。‘arpack’和‘randomized’的适用场景类似，区别是‘randomized’使用的是sklearn自己的svd实现，而arpark直接使用了SciPy库的sparse SVD实现。‘auto’会自动在上述三种中权衡，选择一个合适的SVD算法来降维。默认‘auto’。

In [190]:
```
from sklearn.decomposition import PCA
model = PCA(8).fit(train_data_t)
train_data_pca = model.transform(train_data_t)
train_data_pca
```

Out[190]:
```
array([[ 1.1392773 , -0.29106677, -0.0422555 , ..., -0.01207913,
        -0.06202504, -0.00164481],
       [-0.27083521,  0.00478977, -0.14692378, ...,  0.1556708 ,
        -0.42779591, -0.07282012],
       [-0.04046371,  0.22419335, -0.22882044, ..., -0.03760329,
        -0.1562094 , -0.08777388],
       ...,
       [ 0.6040624 , -0.50130066,  0.03900482, ..., -0.08496466,
         0.29237276,  0.08875735],
       [ 0.7100578 , -0.34696803,  0.0722469 , ..., -0.20182422,
         0.11931491, -0.07102299],
       [ 0.09529519,  0.43108747, -0.03305905, ...,  0.0368424 ,
         0.16884611, -0.01344431]])
```

In [191]:
```
train_data_pca.shape
```

Out[191]:
```
(404, 8)
```