DummyRegressor类的属性：

```
    constant_ : float or array of shape [n_outputs]
    n_outputs_ : int,
    outputs_2d_ : bool,
```

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LinearRegression().fit(boston.data, boston.target)
reg.score(boston.data, boston.target)
```

In [ ]:

```
from sklearn.dummy import DummyRegressor

reg = DummyRegressor().fit(boston.data, boston.target)
reg.score(boston.data, boston.target)
```

7.5 实战练习

以均数为标准将boston数据的因变量拆分为二分类，分别拟合不同的分类预测模型，并进行完整的模型效果评价。

思考在身边可能遇到的聚类分析案例中，有无可能出现ground truth class的情形。

8 数据的拆分

使用建模数据的结果进行模型效果评价，很难避免过拟合发生。

8.1 二分法拆分

sklearn.model_selection.train_test_split(

```
    *arrays : 等长度的需要拆分的数据对象
        格式可以是lists, numpy arrays, scipy稀疏矩阵或者pandas数据框
        显然，对于有监督类模型，X和y需要按相同标准同时进行拆分
    test_size = 0.25 : float, int, None, 用于验证模型的样本比例，范围在0~1
        为None时所有样本都将用于训练
    train_size = None : float, int, or None, 用于训练模型的样本比例，0~1
        为None时自动基于test_size计算
    random_state = None
    shuffle = True : 是否在拆分前对样本做随机排列
    stratify = None : array-like or None，是否按指定类别标签对数据做分层拆分
```

)返回 : 对输入对象进行拆分后的list, length = 2 * len(arrays)

In [ ]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(boston.data,
    boston.target, test_size = 0.3, random_state = 111)
```

In [ ]:

```
len(X_train), len(X_test), len(y_train), len(y_test)
```

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LinearRegression().fit(X_train, y_train)
```

In [ ]:

```
from sklearn.metrics import r2_score

print(r2_score(y_train, reg.predict(X_train)))
print(r2_score(y_test, reg.predict(X_test)))
```

8.2 交叉验证

sklearn中和交叉验证相关的样本拆分函数：

```
model_selection.GroupKFold([n_splits])
model_selection.GroupShuffleSplit([…])
model_selection.KFold([n_splits, shuffle, …])
model_selection.LeaveOneGroupOut()
model_selection.LeavePGroupsOut(n_groups)
model_selection.LeaveOneOut()
model_selection.LeavePOut(p)
model_selection.PredefinedSplit(test_fold)
model_selection.RepeatedKFold([n_splits, …])
model_selection.RepeatedStratifiedKFold([…])
model_selection.ShuffleSplit([n_splits, …])
model_selection.StratifiedKFold([n_splits, …])
model_selection.StratifiedShuffleSplit([…])
model_selection.TimeSeriesSplit([n_splits, …])
```

sklearn中将具体方法与交叉验证直接结合的函数（部分）：

```
linear_model.ElasticNetCV([l1_ratio, eps, …])
linear_model.LarsCV([fit_intercept, …])
linear_model.LassoCV([eps, n_alphas, …])
linear_model.LassoLarsCV([fit_intercept, …])
linear_model.LogisticRegressionCV([Cs, …])
linear_model.RidgeClassifierCV([alphas, …])
linear_model.RidgeCV([alphas, …])
```

8.2.1 将拆分与评价合并执行

sklearn.model_selection.cross_val_score(

```
    estimator : 用于拟合数据的估计器对象名称
    X : array-like, 用于拟合模型的数据阵
    y = None : array-like, 有监督模型使用的因变量
    groups = None : array-like, 形如(n_samples,), 样本拆分时使用的分组标签
    scoring = None : string, callable or None, 模型评分的计算方法
    cv = None : int, 设定交互验证时的样本拆分策略
        None, 使用默认的3组拆分
        integer, 设定具体的拆分组数
        object / iterable 用于设定拆分
    n_jobs = 1, verbose = 0, fit_params = None
    pre_dispatch = '2*n_jobs'
```

)返回：每轮模型对应评分的数组

In [ ]:

```
from sklearn.model_selection import cross_val_score

reg = linear_model.LinearRegression()
scores = cross_val_score(reg, boston.data, boston.target, cv = 10)
scores
```

In [ ]:

```
scores.mean(), scores.std()
```

In [ ]:

```
scores = cross_val_score(reg, boston.data, boston.target,
                         scoring = 'explained_variance', cv = 10)
print(scores.mean())
scores
```

保证案例顺序的随机性

样本中案例顺序如果非随机，将会对模型验证带来严重的影响。

KFold等函数有一个内置的参数shuffle，可以要求在拆分数据前将数据索引随机排序（但该参数默认为False）。
cross_val_score等函数无此参数，因此必要时应当先对数据进行随机排序。

In [ ]:

```
# 对数据进行随机重排，保证拆分的均匀性
import numpy as np

X, y = boston.data, boston.target
indices = np.arange(y.shape[0])
np.random.shuffle(indices)
X, y = X[indices], y[indices]
```

In [ ]:

```
from sklearn.model_selection import cross_val_score

reg = linear_model.LinearRegression()
scores = cross_val_score(reg, X, y, cv = 10)
scores
```

In [ ]:

```
scores.mean(), scores.std()
```

8.2.2 同时使用多个评价指标

cross_validate函数使用的参数基本和cross_val_score相同，但是功能上有以下扩展：

可以指定多个指标对模型进行评估。
除测试集得分之外，还会返回一个包含训练得分，拟合次数，得分次数的字典。

sklearn.model_selection.cross_validate(

```
estimator : 用于拟合数据的估计器对象名称
X : array-like，用于拟合模型的数据阵
y = None : array-like，有监督模型使用的因变量
groups = None : array-like, 形如(n_samples,)，样本拆分时使用的分组标签
scoring = None : string, callable, list/tuple, dict or None
    模型评分的计算方法，多评估指标时使用list/dict等方式提供
cv = None : int，设定交互验证时的样本拆分策略
    None，使用默认的3组拆分
    integer，设定具体的拆分组数
    object / iterable 用于设定拆分
n_jobs = 1, verbose = 0, fit_params = None
pre_dispatch = '2*n_jobs'
return_train_score = True : boolean，是否返回训练集评分
```

)返回：每轮模型对应评分的字典，shape = (n_splits,)

```
test_score : 测试集评分数组
train_score : 训练集评分数组
fit_time : 每轮训练集模型拟合使用的时间
score_time : 每轮测试集模型评分使用的时间
```

In [ ]:

```
from sklearn.model_selection import cross_validate

scoring = ['r2', 'explained_variance']
scores = cross_validate(reg, boston.data, boston.target, cv = 5,
                        scoring = scoring, return_train_score = False)
```

In [ ]:

```
scores
```

In [ ]:

```
scores['test_r2'].mean()
```

8.2.3 使用交互验证后的模型进行预测

sklearn.model_selection.cross_val_predict(

```
    estimator, X, y = None, groups = None, cv = None
    n_jobs = 1, verbose = 0, fit_params = None
    pre_dispatch = '2*n_jobs'
    method = 'predict' : 指明估计器使用的预测命令
        method = 'predict_proba'时，各列按照升序对应各个类别
```

)返回：ndarray，模型对应的各案例预测值

In [ ]:

```
from sklearn.model_selection import cross_val_predict

pred = cross_val_predict(reg, boston.data, boston.target, cv = 5)
pred[:10]
```

In [ ]:

```
# 模型评估结果会和上面有所不同
r2_score(boston.target, pred)
```

8.3 实战练习

自行尝试使用model_selection.KFold()类对iris数据进行拆分，并完成随后的建模分析和评估工作。

```
    官方API文档：http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
```

自行尝试使用linear_model.LogisticRegressionCV()类对iris数据进行交互验证分析。

```
    官方API文档：http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html
```

9 模型参数优化

9.1 网格搜索

class sklearn.model_selection.GridSearchCV(