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(

estimator : 考虑优化的估计器
param_grid : dict or list of dictionaries, 希望进行搜索的参数阵
scoring = None : string/callable/list/tuple/dict/None, 模型评分方法

fit_params = None, n_jobs = 1, cv = None
iid = True : 数据是否在各fold间均匀分布，此时将直接最小化总样本的损失函数
refit = True : 是否使用发现的最佳参数重新拟合估计器
verbose = 0, pre_dispatch = '2*n_jobs', error_score = 'raise'
return_train_score = True : 是否返回训练集的评分

)

GridSearchCV类的属性：

cv_results_ : 字典格式的参数列表，可被直接转换为pandas数据框
best_estimator_ : 网格搜索得出的最佳模型
best_score_ : 最佳模型的平均交互验证得分
best_params_ : dict，最佳模型的参数设定
best_index_ : int，最佳模型对应的索引值
scorer_ : function or a dict，用于选择最佳模型的评分函数
n_splits_ : int，交叉验证的拆分数

GridSearchCV类的方法：

decision_function(*args, **kwargs) : 调用筛选出的最佳模型并返回预测结果
其余标准API接口函数

In [ ]:

```
from sklearn import svm, datasets
from sklearn.model_selection import GridSearchCV

parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svc = svm.SVC(probability = True)

clf = GridSearchCV(svc, parameters)
clf.fit(iris.data, iris.target)
```

In [ ]:

```
# 显示所有拟合模型的参数设定
pd.DataFrame(clf.cv_results_)
```

In [ ]:

```
clf.best_estimator_
```

In [ ]:

```
clf.decision_function(iris.data[:10])
```

In [ ]:

```
clf.predict_proba(iris.data[:10])
```

In [ ]:

```
clf.best_estimator_.predict_proba(iris.data[:10])
```

9.2 随机搜索

在不明确可能的参数候选值时，可以在指定的参数值分布中进行取样，实现对参数的随机搜索。

class sklearn.model_selection.RandomizedSearchCV(

```
    estimator :
    param_distributions : dict, 希望进行搜索的参数字典
    n_iter = 10 : int, 考虑抽取出的参数组合数，该参数用于控制计算总量

    scoring = None, fit_params = None, n_jobs = 1, iid = True

    refit = True : 使用整个数据集重新fit搜索到的最佳参数组合模型
    cv = None, verbose = 0
    pre_dispatch = '2*n_jobs', random_state = None
    error_score = 'raise', return_train_score = True
```

)

RandomizedSearchCV类的属性（和GridSearchCV类相同）：

```
    cv_results_ : 字典格式的参数列表，可被直接转换为pandas数据框
    best_estimator_ : 网格搜索得出的最佳模型
    best_score_ : 最佳模型的平均交互验证得分
    best_params_ : dict，最佳模型的参数设定
    best_index_ : int，最佳模型对应的索引值
    scorer_ : function or a dict，用于选择最佳模型的评分函数
    n_splits_ : int，交叉验证的拆分数
```

RandomizedSearchCV类的方法（和GridSearchCV类相同）：

```
    decision_function(*args, **kwargs) : 调用最佳模型，并返回预测结果
    其余标准API接口函数
```

In [ ]:

```
import scipy.stats as ss
from sklearn import svm, datasets
from sklearn.model_selection import RandomizedSearchCV

iris = datasets.load_iris()
# parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
parameters = {'C': ss.expon(scale = 100),
              'gamma': ss.expon(scale = .1),
              'kernel': ['linear', 'rbf'],
              'class_weight':['balanced', None]}
svc = svm.SVC()

clf = RandomizedSearchCV(svc, parameters)
clf.fit(iris.data, iris.target)
```

In [ ]:

```
# 显示所有拟合模型的参数设定
pd.DataFrame(clf.cv_results_)
```

In [ ]:

```
clf.best_estimator_
```

In [ ]:

```
clf.decision_function(iris.data)[:10]
```

9.3 验证曲线

结合网格搜索等参数调优方法，对模型优化的效果进行图形化的评价。

由于训练集的评分已经被用于参数调优，因此该评分用于评估效果已经不再客观，需要使用验证集的评分用于评估。

sklearn.model_selection.validation_curve(

```
estimator, X, y

param_name : string, 在模型搜索中需要改变的参数
param_range : array-like, shape (n_values,), 相应参数的具体值列表

groups = None : array-like, with shape (n_samples,)
    拆分为训练/测试集时用到的样本分组标签
cv = None : int, 交互验证时的分组方法，为None时按照三组拆分

scoring = None, n_jobs = 1, pre_dispatch = 'all', verbose = 0
```

)返回:

```
train_scores : array, shape (n_ticks, n_cv_folds), 测试集的模型评分
test_scores : array, shape (n_ticks, n_cv_folds), 验证集的模型评分
```

In [ ]:

```
import numpy as np
from sklearn.model_selection import validation_curve
from sklearn.linear_model import Ridge

# 将原始数据打乱为随机顺序
np.random.seed(0)
X, y = boston.data, boston.target
indices = np.arange(y.shape[0])
np.random.shuffle(indices)
X, y = X[indices], y[indices]

# 返回评价结果
train_scores, test_scores = validation_curve(Ridge(), X, y, "alpha",
                                             np.logspace(-7, 3, 30))
```

In [ ]:

```
print(len(test_scores))
test_scores[:5]
```

In [ ]:

```
np.mean(test_scores, axis = 1)
```

In [ ]:

```
plt.scatter(np.logspace(-7, 3, 30), np.mean(test_scores, axis = 1))
```

9.4 学习曲线

学习曲线用于评估多大的样本量用于训练才能达到最佳效果。

sklearn.model_selection.learning_curve(

```
    estimator, X, y, groups = None
    train_sizes = array([ 0.1, 0.325, 0.55, 0.775, 1. ]) :
        模型拟合时用于训练集的相对/绝对样本数，用整数表示绝对样本数

    cv = None, scoring = None
    exploit_incremental_learning = False : 是否使用增量学习策略
    n_jobs = 1, pre_dispatch = 'all', verbose = 0
    shuffle = False, random_state = None
```

)返回:

```
    train_sizes_abs : array, shape = (n_unique_ticks,), 训练集大小
    train_scores : array, shape (n_ticks, n_cv_folds), 训练集评分
    test_scores : array, shape (n_ticks, n_cv_folds), 验证集评分
```

In [ ]:

```
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC

# 将原始数据打乱为随机顺序
np.random.seed(0)
X, y = iris.data, iris.target
indices = np.arange(y.shape[0])
np.random.shuffle(indices)
X, y = X[indices], y[indices]

size = [30, 50, 70, 90, 110, 120] # 注意这里不能设为150

train_sizes, train_scores, test_scores = learning_curve(
    SVC(kernel='linear'), X, y, train_sizes = size, cv = 5)
```

In [ ]:

```
train_scores
```

In [ ]:

```
test_scores
```

In [ ]:

```
np.mean(train_scores, axis = 1)
```

In [ ]:

```
# 用散点图或者线图的方式来绘制曲线
plt.scatter(y = np.mean(train_scores, axis = 1), x = size)
plt.scatter(y = np.mean(test_scores, axis = 1), x = size)
```

In [ ]:

```
# 用散点图或者线图的方式来绘制曲线
sns.lineplot(y = np.mean(train_scores, axis = 1), x = size)
sns.lineplot(y = np.mean(test_scores, axis = 1), x = size)
```

In [ ]:

```
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC

# 将原始数据打乱为随机顺序
X, y = boston.data, boston.target
indices = np.arange(y.shape[0])
np.random.shuffle(indices)
X, y = X[indices], y[indices]

size = np.linspace(0.1, 1, 10) # 以百分比的形式设定样本量

from sklearn import linear_model

reg = linear_model.LinearRegression()

train_sizes, train_scores, test_scores = learning_curve(
    reg, X, y, train_sizes = size, cv = 5)
train_scores
```

In [ ]:

```
plt.scatter(y = np.mean(train_scores, axis = 1), x = train_sizes)
plt.scatter(y = np.mean(test_scores, axis = 1), x = train_sizes)
```

9.5 实战练习

使用网格搜索功能重新拟合岭回归数据，比较两种实现方式的异同，包括分析结果、所需时间等。

当同时对多个参数进行调优时，该如何实现绘制验证曲线的功能？请思考解决方案，并用程序加以实现。

10 模型集成