

模型评价与验证

项目 1: 预测波士顿房价

--- ## 第一步. 导入数据

在这个项目中，你将利用马萨诸塞州波士顿郊区的房屋信息数据训练和测试一个模型，并对模型的性能和预测能力进行测试。通过该数据训练后的好的模型可以被用来对房屋做特定预测---尤其是对房屋的价值。对于房地产经纪等人的日常工作来说，这样的预测模型被证明非常有价值。

此项目的数据集来自[UCI机器学习知识库\(数据集已下线\)1](https://archive.ics.uci.edu/ml/datasets.html)
(<https://archive.ics.uci.edu/ml/datasets.html>)。

原始数据：

每个类的观察值数量是均等的，共有 506 个观察，13 个输入变量和1个输出变量。变量名如下：

CRIM: 城镇人均犯罪率。

ZN: 住宅用地超过 25000 sq.ft. 的比例。

INDUS: 城镇非零售商用土地的比例。

CHAS: 查理斯河空变量（如果边界是河流，则为1；否则为0）。

NOX: 一氧化氮浓度。

RM: 住宅平均房间数。

AGE: 1940 年之前建成的自用房屋比例。

DIS: 到波士顿五个中心区域的加权距离。

RAD: 辐射性公路的接近指数。

TAX: 每 10000 美元的全值财产税率。

PTRATIO: 城镇师生比例。

B: 1000 (Bk-0.63) ^ 2, 其中 Bk 指代城镇中黑人的比例。

LSTAT: 人口中地位低下者的比例。

MEDV: 自住房的平均房价，以千美元计。

波士顿房屋这些数据于1978年开始统计，共506个数据点，涵盖了麻省波士顿不同郊区房屋14种特征的信息。

本项目对原始数据集做了以下处理：

- 有16个`'MEDV'` 值为50.0的数据点被移除。 这很可能是由于这些数据点包含**遗失**或**看不到的值**。
- 有1个数据点的 `'RM'` 值为8.78. 这是一个异常值，已经被移除。
- 对于本项目，房屋的`'RM'`，`'LSTAT'`，`'PTRATIO'` 以及`'MEDV'` 特征是必要的，其余不相关特征已经被移除。
- `'MEDV'` 特征的值已经过必要的数学转换，可以反映35年来市场的通货膨胀效应。

运行下面区域的代码以载入波士顿房屋数据集，以及一些此项目所需的Python库。如果成功返回数据集的大小，表示数据集已载入成功。

```
In [32]: # 载入此项目所需要的库
import numpy as np
import pandas as pd
import visuals as vs # Supplementary code

# 让结果在notebook中显示
%matplotlib inline
```

```
In [33]: # 载入波士顿房屋的数据集
data = pd.read_csv('housing.csv')
prices = data['MEDV']
features = data.drop('MEDV', axis = 1)

# 完成
print "Boston housing dataset has {} data points with {} variables each.".format(
    len(data), len(data.columns))

Boston housing dataset has 489 data points with 4 variables each.
```

第二步. 分析数据

在项目的第一部分，你会对波士顿房地产数据进行初步的观察并给出你的分析。通过对数据的探索来熟悉数据可以让你更好地理解和解释你的结果。

由于这个项目的最终目标是建立一个预测房屋价值的模型，我们需要将数据集分为**特征(features)**和**目标变量(target variable)**。

- 特征 'RM', 'LSTAT', 和 'PTRATIO'，给我们提供了每个数据点的数量相关的信息。
- 目标变量： 'MEDV'，是我们希望预测的变量。

他们分别被存在**features**和**prices**两个变量名中。

编程练习 1：基础统计运算

你的第一个编程练习是计算有关波士顿房价的描述统计数据。我们已为你导入了**numpy**，你需要使用这个库来执行必要的计算。这些统计数据对于分析模型的预测结果非常重要的。在下面的代码中，你要做的是：

- 计算**prices**中的'MEDV'的最小值、最大值、均值、中值和标准差；
- 将运算结果储存在相应的变量中。

In [34]: #TODO 1

```
#目标: 计算价值的最小值
minimum_price = np.min(prices)

#目标: 计算价值的最大值
maximum_price = np.max(prices)

#目标: 计算价值的平均值
mean_price = np.mean(prices)

#目标: 计算价值的中值
median_price = np.median(prices)

#目标: 计算价值的标准差
std_price = np.std(prices)

#目标: 输出计算的结果
print "Statistics for Boston housing dataset:\n"
print "Minimum price: ${:,.2f}".format(minimum_price)
print "Maximum price: ${:,.2f}".format(maximum_price)
print "Mean price: ${:,.2f}".format(mean_price)
print "Median price ${:,.2f}".format(median_price)
print "Standard deviation of prices: ${:,.2f}".format(std_price)
```

Statistics for Boston housing dataset:

```
Minimum price: $105,000.00
Maximum price: $1,024,800.00
Mean price: $454,342.94
Median price $438,900.00
Standard deviation of prices: $165,171.13
```

问题 1 - 特征观察

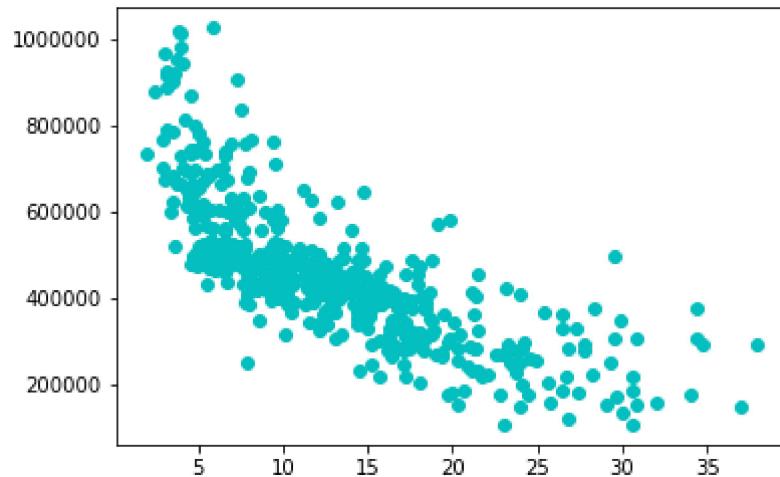
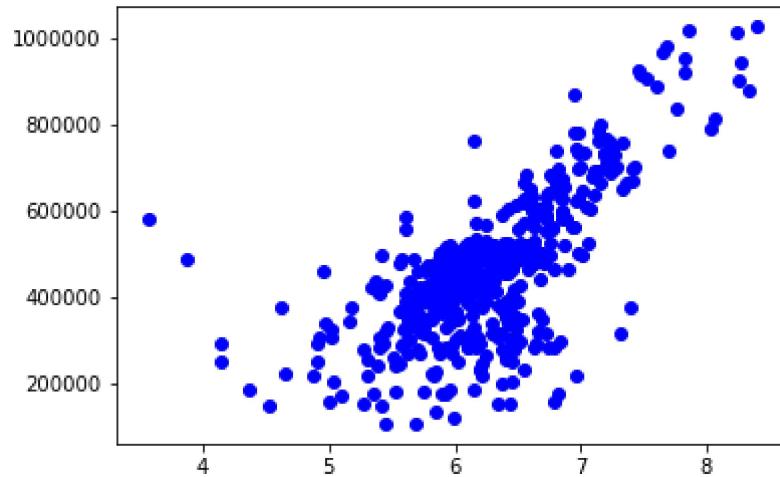
如前文所述，本项目中我们关注的是其中三个值：'RM'、'LSTAT' 和 'PTRATIO'，对每一个数据点：

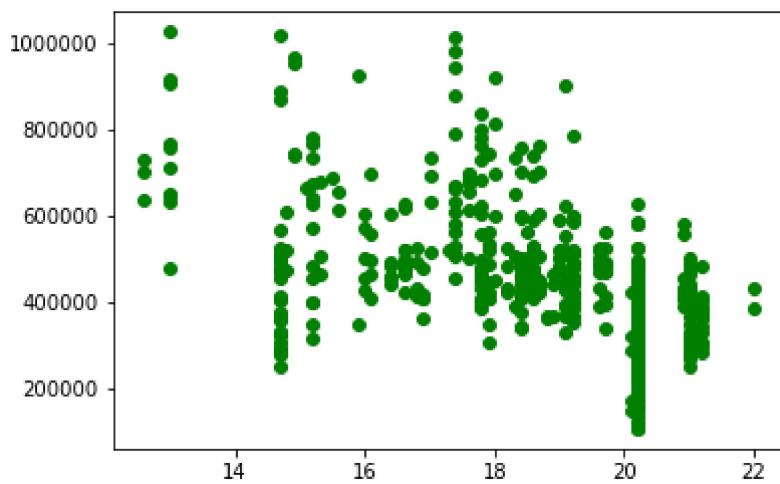
- 'RM' 是该地区中每个房屋的平均房间数量；
- 'LSTAT' 是指该地区有多少百分比的房东属于低收入阶层（有工作但收入微薄）；
- 'PTRATIO' 是该地区的中学和小学里，学生和老师的数目比（学生/老师）。

凭直觉，上述三个特征中对每一个来说，你认为增大该特征的数值，'MEDV' 的值会是增大还是减小呢？每一个答案都需要你给出理由。

提示：你预期一个 'RM' 值是6的房屋跟 'RM' 值是7的房屋相比，价值更高还是更低呢？

```
In [35]: import matplotlib
rm = data['RM']
medv = data['MEDV']
matplotlib.pyplot.scatter(rm, medv, c='b')
matplotlib.pyplot.show()
lstat = data['LSTAT']
matplotlib.pyplot.scatter(lstat, medv, c='c')
matplotlib.pyplot.show()
ptratio = data['PTRATIO']
matplotlib.pyplot.scatter(ptratio, medv, c='g')
matplotlib.pyplot.show()
```





问题 1 - 回答:

'RM': 增大该值，'MEDV'的值会增大，因为这意味着房屋面积，硬件设施等的增加，以及更高的经济水平。

'LSTAT': 增大该值，'MEDV'的值会减小，因为低收入阶层百分比增大意味着该区域经济水平较低。

'PTRATIO': 增大该值，'MEDV'的值不会发生明显的变化，因为学生/老师的数目比体现了该地区的教育资源配置，该值提高会增加学区房价格，但是并不会对中位数造成明显影响。

根据散点图也可看出数据趋势。

编程练习 2: 数据分割与重排

接下来，你需要把波士顿房屋数据集分成训练和测试两个子集。通常在这个过程中，数据也会被重排列，以消除数据集中由于顺序而产生的偏差。在下面的代码中，你需要

使用 `sklearn.model_selection` 中的 `train_test_split`，将 `features` 和 `prices` 的数据都分成用于训练的数据子集和用于测试的数据子集。

- 分割比例为：80%的数据用于训练，20%用于测试；
- 选定一个数值以设定 `train_test_split` 中的 `random_state`，这会确保结果的一致性；

In [36]: # TODO 2

```
# 提示: 导入train_test_split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, prices, test_size=0)

print("Train test split success!")
```

Train test split success!

问题 2 - 训练及测试

将数据集按一定比例分为训练用的数据集和测试用的数据集对学习算法有什么好处？

如果用模型已经见过的数据，例如部分训练集数据进行测试，又有什么坏处？

提示：如果没有数据来对模型进行测试，会出现什么问题？

问题 2 - 回答：

1. 将数据集按一定比例分成两部分可以用测试集验证评估模型对新数据的预测效果，比如是否存在欠拟合和过拟合问题等。 2. 如果用模型已经见过的数据，那么体现不出模型的泛化能力，即对新数据的预测效果。 3. 如果没有测试集则我们对模型的好坏无法作出准确的评估。

第三步. 模型衡量标准

在项目的第三步中，你需要了解必要的工具和技巧来让你的模型进行预测。用这些工具和技巧对每一个模型的表现做精确的衡量可以极大地增强你预测的信心。

编程练习3：定义衡量标准

如果不能对模型的训练和测试的表现进行量化地评估，我们就很难衡量模型的好坏。通常我们会定义一些衡量标准，这些标准可以通过对某些误差或者拟合程度的计算来得到。在这个项目中，你将通过运算决定系数 (http://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination) R^2 来量化模型的表现。模型的决定系数是回归分析中十分常用的统计信息，经常被当作衡量模型预测能力好坏的标准。

R^2 的数值范围从0至1，表示目标变量的预测值和实际值之间的相关程度平方的百分比。一个模型的 R^2 值为0还不如直接用平均值来预测效果好；而一个 R^2 值为1的模型则可以对目标变量进行完美的预测。从0至1之间的数值，则表示该模型中目标变量中有百分之多少能够用特征来解释。模型也可能出现负值的 R^2

在下方代码的 `performance_metric` 函数中，你要实现：

- 使用 `sklearn.metrics` 中的 `r2_score` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html) 来计算 `y_true` 和 `y_predict` 的 R^2 值，作为对其表现的评判。
- 将他们的表现评分储存到 `score` 变量中。

或

- (可选) 不使用任何外部库，参考决定系数的定义 (https://en.wikipedia.org/wiki/Coefficient_of_determination) 进行计算，这也可以帮助你更好的理解决定系数在什么情况下等于0或等于1。

In [37]: # TODO 3

```
# 提示: 导入r2_score
from sklearn.metrics import r2_score

def performance_metric(y_true, y_predict):
    """计算并返回预测值相比于预测值的分数"""

    score = r2_score(y_true, y_predict, sample_weight=None, multioutput=None)

    return score
```

In [38]: # TODO 3 可选

```
# 不允许导入任何计算决定系数的库

def performance_metric2(y_true, y_predict):
    """计算并返回预测值相比于预测值的分数"""

    score = None

    return score
```

问题 3 - 拟合程度

假设一个数据集有五个数据且一个模型做出下列目标变量的预测：

真实数值	预测数值
3.0	2.5
-0.5	0.0
2.0	2.1
7.0	7.8
4.2	5.3

你觉得这个模型已成功地描述了目标变量的变化吗？如果成功，请解释为什么，如果没有，也请给出原因。

提示：运行下方的代码，使用performance_metric函数来计算模型的决定系数。

In [39]: # 计算这个模型的预测结果的决定系数

```
score = performance_metric([3, -0.5, 2, 7, 4.2], [2.5, 0.0, 2.1, 7.8, 5.3])
print "Model has a coefficient of determination, R^2, of {:.3f}.".format(score)
```

Model has a coefficient of determination, R^2, of 0.923.

问题 3 - 回答：

Model has a coefficient of determination, R^2, of 0.923.

一个R²值为1的模型则可以对目标变量进行完美的预测。从0至1之间的数值，则表示该模型中目标变量中有百分之多少能够用特征来解释。目标变量中有92.3%能够用特征来解释，因此模型成功地描述了目标变量的变化。

第四步. 分析模型的表现

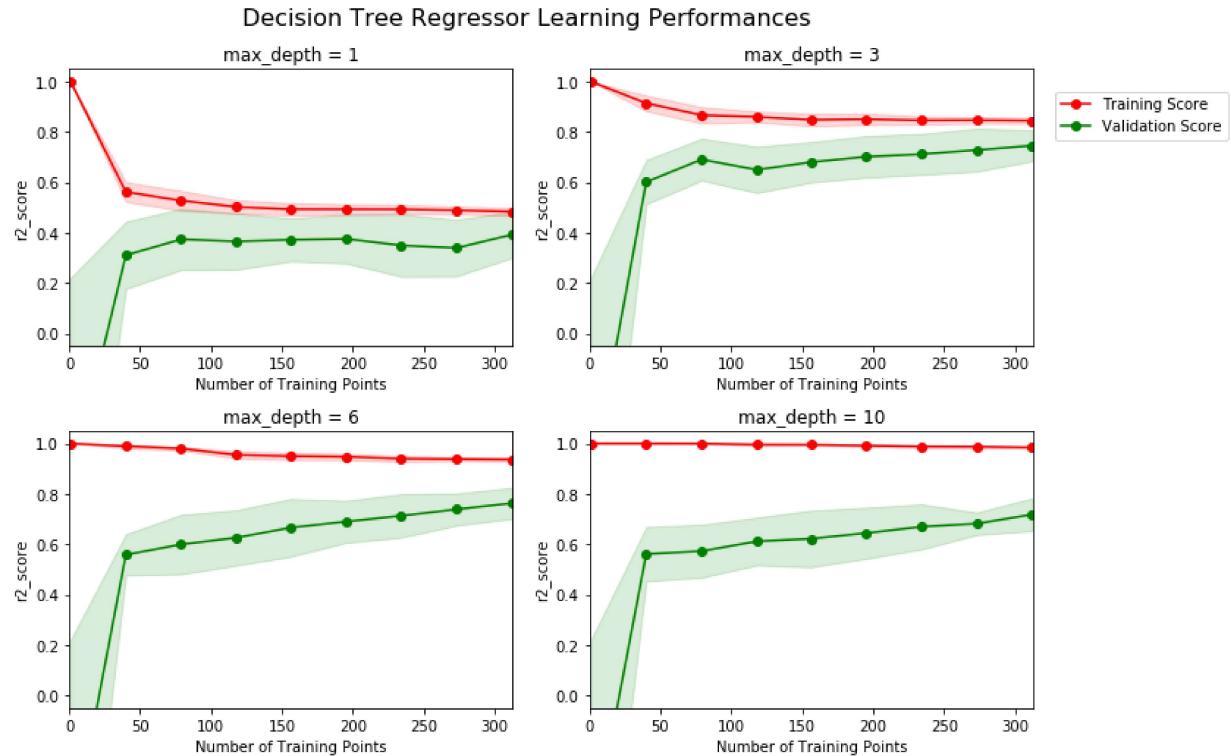
在项目的第四步，我们来看一下不同参数下，模型在训练集和验证集上的表现。这里，我们专注于一个特定的算法（带剪枝的决策树，但这并不是这个项目的特点），和这个算法的一个参数'`max_depth'`。用全部训练集训练，选择不同'`max_depth`'参数，观察这一参数的变化如何影响模型的表现。画出模型的表现来对于分析过程十分有益，这可以让我们看到一些单看结果看不到的行为。

学习曲线

下方区域内的代码会输出四幅图像，它们是一个决策树模型在不同最大深度下的表现。每一条曲线都直观得显示了随着训练数据量的增加，模型学习曲线的在训练集评分和验证集评分的变化，评分使用决定系数R²。曲线的阴影区域代表的是该曲线的不确定性（用标准差衡量）。

运行下方区域中的代码，并利用输出的图形回答下面的问题。

```
In [40]: # 根据不同的训练集大小, 和最大深度, 生成学习曲线
vs.ModelLearning(X_train, y_train)
```



问题 4 - 学习曲线

选择上述图像中的其中一个，并给出其最大深度。随着训练数据量的增加，训练集曲线的评分有怎样的变化？验证集曲线呢？如果有更多的训练数据，是否能有效提升模型的表现呢？

提示：学习曲线的评分是否最终会收敛到特定的值？

问题 4 - 回答：

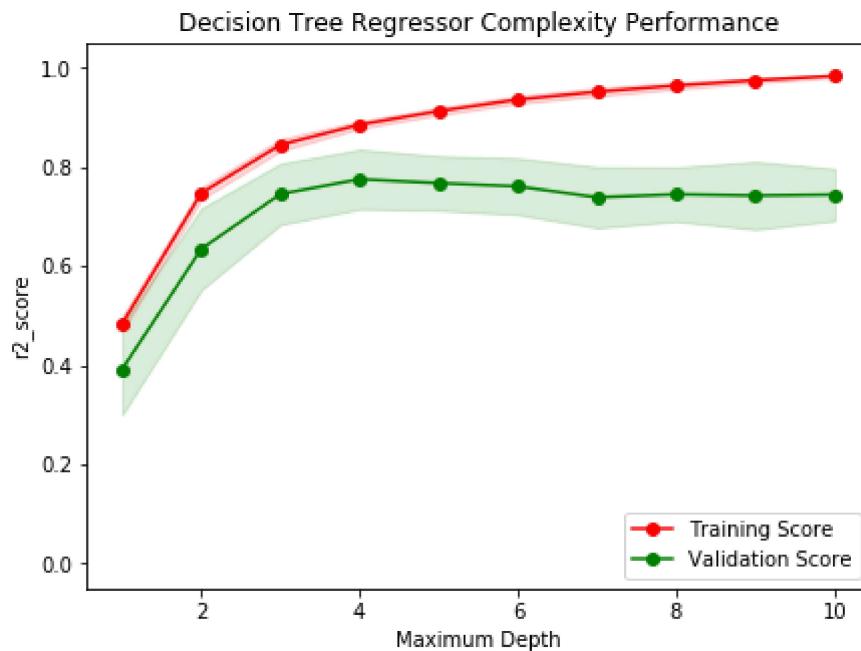
对于 `max_depth = 3` 的图像来说，其最大深度为 3。随着训练数据量的增加，训练集曲线的评分逐渐下降，最终趋于稳定；验证集曲线的评分逐渐上升，最终也趋于稳定。如果有更多的训练数据，模型的表现不会提升太多。从图中看关键与最大深度有关，随其提高，训练集曲线评分和测试集曲线评分都得到了提高。图中最大深度为 3 时两条曲线最终收敛到的评分都很高，而对于最大深度为 10 的图像，由于深度过深发生了过拟合，训练集的得分很高接近 100%，而测试集则得分很低。

复杂度曲线

下列代码内的区域会输出一幅图像，它展示了一个已经经过训练和验证的决策树模型在不同最大深度条件下的表现。这个图形将包含两条曲线，一个是训练集的变化，一个是验证集的变化。跟学习曲线相似，阴影区域代表该曲线的不确定性，模型训练和测试部分的评分都用的 `performance_metric` 函数。

运行下方区域中的代码，并利用输出的图形并回答下面的两个问题。

```
In [41]: # 根据不同的最大深度参数，生成复杂度曲线
vs.ModelComplexity(X_train, y_train)
```



问题 5 - 偏差（bias）与方差（variance）之间的权衡取舍

当模型以最大深度 1 训练时，模型的预测是出现很大的偏差还是出现了很大的方差？当模型以最大深度 10 训练时，情形又如何呢？图形中的哪些特征能够支持你的结论？

提示：你如何得知模型是否出现了偏差很大或者方差很大的问题？

问题 5 - 回答：

1.当模型以最大深度1训练时，模型的预测是出现了很大的偏差，欠拟合。 2.当模型以最大深度10训练时，出现了很大的方差，即过拟合，由于模型过于复杂，对于训练集匹配度很高，但是泛化能力不足，测试集得分降低 3.通过增大数据量和选择适合的模型复杂度可以改善效果。

问题 6- 最优模型的猜测

结合问题 5 中的图，你认为最大深度是多少的模型能够最好地对未见过的数据进行预测？你得出这个答案的依据是什么？

问题 6 - 回答：

最大深度为4的模型

依据：当最大深度为4时，测试集的得分达到最高值。当最大深度大于4时，虽然训练集的得分逐步提高但是测试集的得分开始下降。

第五步. 选择最优参数

问题 7- 网格搜索（Grid Search）

什么是网格搜索法？如何用它来优化模型？

问题 7 - 回答：

网格搜索法是指定参数值的一种穷举搜索方法，通过将估计函数的参数通过交叉验证的方法进行优化来得到最优的学习算法。即将各个参数可能的取值进行排列组合，列出所有可能的组合结果生成“网格”。然后将各组合用于SVM训练，使用交叉验证对表现进行评估。在拟合函数尝试了所有的参数组合后，返回一个合适的分类器，自动调整至最佳参数组合。

问题 8 - 交叉验证

- 什么是K折交叉验证法（k-fold cross-validation）？
- [GridSearchCV \(http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html\)](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)是如何结合交叉验证来完成对最佳参数组合的选择的？
- [GridSearchCV \(http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html\)](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)中的'cv_results_'属性能告诉我们什么？

- 网格搜索时如果不使用交叉验证会有什么问题？交叉验证又是如何解决这个问题的？

提示：在下面 `fit_model` 函数最后加入 `print pd.DataFrame(grid.cv_results_)` 可以帮你查看更多信息。

问题 8 - 回答：

1.k折交叉验证法（k-fold cross-validation）即先将全部数据集按照一定比例分成训练集和测试集，然后将训练集分成k个子集，每个子集均做一次测试集，其余的作为训练集。如此交叉验证重复k次，每次选择一个子集作为测试集，并将k次的平均交叉验证识别率作为结果。优点：所有的样本都被作为了训练集和测试集，每个样本都被验证了一次。切分方式：默认情况下Kfold是对数据按顺序切分。

2.网格搜索算法（GridSearchCV）用于系统地遍历多种参数的组合，通过交叉验证来确定最佳效果参数。

3.GridSearchCV中的'cvresults'能够输出一个dict，其中包括相关参数的名称以及对应的值。

4.a.网格搜索时如果不使用交叉验证则可能使得模型泛化能力无法达到最优：因为网格搜索就是尝试各种可能的参数组合值，然后进行交叉验证，找出使交叉验证精确度最高的参数组合。如果不使用交叉验证，那么对于找到的好参数，不能确保其最精确地预测未知的数据。 b.像对数据进行单次分割来进行网格搜索，可能会有什么问题？单次分割不能反映整体预测水平，可能存在该单次分割预测效果好而其他分割情况下预测效果较差的情况，不能保证对应的参数组合就是最优组合。 c.交叉验证又是如何避免这个问题的？交叉验证每次选择一个子集作为测试集，并将k次的平均交叉验证识别率作为结果，通过确定最佳效果参数来保证泛化能力。

编程练习 4：训练最优模型

在这个练习中，你将需要将所学到的内容整合，使用决策树算法训练一个模型。为了得出的是一个最优模型，你需要使用网格搜索法训练模型，以找到最佳的 '`max_depth`' 参数。你可以把'`max_depth`' 参数理解为决策树算法在做出预测前，允许其对数据提出问题的数量。决策树是监督学习算法中的一种。

在下方 `fit_model` 函数中，你需要做的是：

1. 定义 '`cross_validator`' 变量：使用 `sklearn.model_selection` 中的 `KFold` (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html) 创建一个交叉验证生成器对象；
2. 定义 '`regressor`' 变量：使用 `sklearn.tree` 中的 `DecisionTreeRegressor` (<http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>) 创建一个决策树的回归函数；
3. 定义 '`params`' 变量：为 '`max_depth`' 参数创造一个字典，它的值是从1至10的数组；
4. 定义 '`scoring_fnc`' 变量：使用 `sklearn.metrics` 中的 `make_scorer` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html) 创建一个评分函数；将 '`performance_metric`' 作为参数传至这个函数中；
5. 定义 '`grid`' 变量：使用 `sklearn.model_selection` 中的 `GridSearchCV` (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) 创建一个网格搜索对象；将变量 '`regressor`', '`params`', '`scoring_fnc`' 和 '`cross_validator`' 作为参数传至这个对象构造函数中；

如果你对python函数的默认参数定义和传递不熟悉，可以参考这个MIT课程的视频 (http://cn-static.udacity.com/mlnd/videos/MIT600XXT114-V004200_DTH.mp4)。

In [42]: # TODO 4

```
# 提示：导入 'KFold' 'DecisionTreeRegressor' 'make_scorer' 'GridSearchCV'
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV

def fit_model(X, y):
    """ 基于输入数据 [X,y]，利于网格搜索找到最优的决策树模型"""

    cross_validator = KFold(n_splits=10, shuffle=False, random_state=None)

    regressor = DecisionTreeRegressor()

    params = {'max_depth':[1,2,3,4,5,6,7,8,9,10]}

    scoring_fnc = make_scorer(performance_metric)

    grid = GridSearchCV(estimator=regressor, param_grid=params, scoring=scoring_fnc)

    # 基于输入数据 [X,y]，进行网格搜索
    grid = grid.fit(X, y)

    # 返回网格搜索后的最优模型
    return grid.best_estimator_
```

编程练习 4：训练最优模型（可选）

在这个练习中，你将需要将所学到的内容整合，使用决策树算法训练一个模型。为了得出的是一个最优模型，你需要使用网格搜索法训练模型，以找到最佳的 'max_depth' 参数。你可以把'max_depth' 参数理解为决策树算法在做出预测前，允许其对数据提出问题的数量。决策树是监督学习算法中的一种。

在下方 `fit_model` 函数中，你需要做的是：

- 遍历参数'max_depth' 的可选值 1~10，构造对应模型
- 计算当前模型的交叉验证分数
- 返回最优交叉验证分数对应的模型

In [43]: # TODO 4 可选

```
...
不允许使用 DecisionTreeRegressor 以外的任何 sklearn 库
提示：你可能需要实现下面的 cross_val_score 函数

def cross_val_score(estimator, X, y, scoring = performance_metric, cv=3):
    """ 返回每组交叉验证的模型分数的数组 """
    scores = [0,0,0]
    return scores
...

def fit_model2(X, y):
    """ 基于输入数据 [X,y]，利于网格搜索找到最优的决策树模型"""

    #最优交叉验证分数对应的最优模型
    best_estimator = None

    return best_estimator
```

问题 9 - 最优模型

最优模型的最大深度 (*maximum depth*) 是多少？此答案与你在问题6所做的猜测是否相同？

运行下方区域内的代码，将决策树回归函数代入训练数据的集合，以得到最优化的模型。

In [44]:

```
# 基于训练数据，获得最优模型
optimal_reg = fit_model(X_train, y_train)

# 输出最优模型的 'max_depth' 参数
print "Parameter 'max_depth' is {} for the optimal model.".format(optimal_reg.get_
Parameter 'max_depth' is 4 for the optimal model.
```

问题 9 - 回答：

Parameter 'max_depth' is 4 for the optimal model. 此答案与我在问题6所做的猜测相同。

第六步. 做出预测

当我们用数据训练出一个模型，它现在就可用于对新的数据进行预测。在决策树回归函数中，模型已经学会对新输入的数据提问，并返回对目标变量的预测值。你可以用这个预测来获取数据未知目标变量的信息，这些数据必须是不包含在训练数据之内的。

问题 10 - 预测销售价格

想像你是一个在波士顿地区的房屋经纪人，并期待使用此模型以帮助你的客户评估他们想出售的房屋。你已经从你的三个客户收集到以下的资讯：

特征	客户 1	客户 2	客户 3
房屋内房间总数	5 间房间	4 间房间	8 间房间
社区贫困指数（%被认为是贫困阶层）	17%	32%	3%
邻近学校的学生-老师比例	15: 1	22: 1	12: 1

你会建议每位客户的房屋销售的价格为多少？从房屋特征的数值判断，这样的价格合理吗？为什么？

提示：用你在分析数据部分计算出来的统计信息来帮助你证明你的答案。

运行下列的代码区域，使用你优化的模型来为每位客户的房屋价值做出预测。

```
In [45]: # 生成三个客户的数据
client_data = [[5, 17, 15], # 客户 1
               [4, 32, 22], # 客户 2
               [8, 3, 12]] # 客户 3

# 进行预测
predicted_price = optimal_reg.predict(client_data)
for i, price in enumerate(predicted_price):
    print "Predicted selling price for Client {}'s home: ${:,.2f}".format(i+1, pr
Predicted selling price for Client 1's home: $391,183.33
Predicted selling price for Client 2's home: $189,123.53
Predicted selling price for Client 3's home: $942,666.67
```

问题 10 - 回答：

Predicted selling price for Client 1's home: \$391,183.33

Predicted selling price for Client 2's home: \$189,123.53

Predicted selling price for Client 3's home: \$942,666.67

Minimum price: \$105,000.00

Maximum price: \$1,024,800.00

Mean price: \$454,342.94

Median price: \$438,900.00

Standard deviation of prices: \$165,171.13

综合三个特征向量以及数据对比，客户3在富人区，房价接近Maximum price；客户2在较贫困区，房价接近Minimum price；客户1房价接近Mean price。模型预测结果与问题1-特征观察的分析一致，是合理的。

编程练习 5

你刚刚预测了三个客户的房子的售价。在这个练习中，你将用你的最优模型在整个测试数据上进行预测，并计算相对于目标变量的决定系数 R^2 的值**。

In [46]: #TODO 5

```
# 提示: 你可能需要用到 X_test, y_test, optimal_reg, performance_metric
# 提示: 你可能需要参考问题10的代码进行预测
predicted_price = optimal_reg.predict(X_test)
# 提示: 你可能需要参考问题3的代码来计算R^2的值
r2 = performance_metric(y_test, predicted_price)

print "Optimal model has R^2 score {:.2f} on test data".format(r2)
```

```
Optimal model has R^2 score 0.74 on test data
```

问题11 - 分析决定系数

你刚刚计算了最优模型在测试集上的决定系数，你会如何评价这个结果？

问题11 - 回答

测试集目标变量中有74%能够用特征来解释，因此模型基本可以描述了目标变量变化。

模型健壮性

一个最优的模型不一定是一个健壮模型。有的时候模型会过于复杂或者过于简单，以致于难以泛化新增添的数据；有的时候模型采用的学习算法并不适用于特定的数据结构；有的时候样本本身可能有太多噪点或样本过少，使得模型无法准确地预测目标变量。这些情况下我们会说模型是欠拟合的。

问题 12 - 模型健壮性

模型是否足够健壮来保证预测的一致性？

提示: 执行下方区域中的代码，采用不同的训练和测试集执行 `fit_model` 函数10次。注意观察对一个特定的客户来说，预测是如何随训练数据的变化而变化的。

In [47]: # 请先注释掉 fit_model 函数里的所有 print 语句
vs.PredictTrials(features, prices, fit_model, client_data)

```
Trial 1: $391,183.33
Trial 2: $411,417.39
Trial 3: $415,800.00
Trial 4: $420,622.22
Trial 5: $423,300.00
Trial 6: $411,931.58
Trial 7: $399,663.16
Trial 8: $407,232.00
Trial 9: $402,531.82
Trial 10: $413,700.00
```

```
Range in prices: $32,116.67
```

问题 12 - 回答：

根据执行结果，预测值浮动在10%以内，健壮性良好。

问题 13 - 实用性探讨

简单地讨论一下你建构的模型能否在现实世界中使用？

提示：回答以下几个问题，并给出相应结论的理由：

- 1978年所采集的数据，在已考虑通货膨胀的前提下，在今天是否仍然适用？
- 数据中呈现的特征是否足够描述一个房屋？
- 在波士顿这样的大都市采集的数据，能否应用在其它乡镇地区？
- 你觉得仅仅凭房屋所在社区的环境来判断房屋价值合理吗？

问题 13 - 回答：

1. 1978年采集的数据在今天已经不再适用，现在与当时经济条件差异巨大。

2. 数据中仅呈现的三个特征太少不足以描述一个房屋。

3. 在波士顿这样的大都市采集的数据不能用于其它乡镇的。由于乡镇和大都市经济差异很大，因此影响各自房价的特征向量会有较大差异。

4. 仅仅凭房屋所在社区的环境来判断房屋价值不合理。社区环境相对于整个都市或者乡镇只是一个很小的缩影，不能反映整个区域的房价水平。

可选问题 - 预测北京房价

(本题结果不影响项目是否通过) 通过上面的实践，相信你对机器学习的一些常用概念有了很好的领悟和掌握。但利用70年代的波士顿房价数据进行建模的确对我们来说意义不是太大。现在你可以把你上面所学应用到北京房价数据集中 `bj_housing.csv`。

免责声明：考虑到北京房价受到宏观经济、政策调整等众多因素的直接影响，预测结果仅供参考。

这个数据集的特征有：

- **Area:** 房屋面积，平方米
- **Room:** 房间数，间
- **Living:** 厅数，间
- **School:** 是否为学区房，0或1
- **Year:** 房屋建造时间，年
- **Floor:** 房屋所处楼层，层

目标变量：

- **Value:** 房屋人民币售价，万

你可以参考上面学到的内容，拿这个数据集来练习数据分割与重排、定义衡量标准、训练模型、评价模型表现、使用网格搜索配合交叉验证对参数进行调优并选出最佳参数，比较两者的差别，最终得出最佳模型对验证集的预测分数。

In [57]: # TODO 6

```
# 你的代码

# 载入此项目所需要的库
import numpy as np
import pandas as pd
import visuals as vs
import matplotlib
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV

def performance_metric(y_true, y_predict):
    """计算并返回预测值相比于预测值的分数"""

    score = r2_score(y_true, y_predict, sample_weight=None, multioutput=None)
    return score

def fit_model(X, y):
    """ 基于输入数据 [X,y], 利于网格搜索找到最优的决策树模型"""

    cross_validator = KFold(n_splits=10, shuffle=False, random_state=None)
    regressor = DecisionTreeRegressor()
    params = {'max_depth':[1,2,3,4,5,6,7,8,9,10]}
    scoring_fnc = make_scorer(performance_metric)
    grid = GridSearchCV(estimator=regressor, param_grid=params, scoring=scoring_fnc)
    # 基于输入数据 [X,y], 进行网格搜索
    grid = grid.fit(X, y)
    # 返回网格搜索后的最优模型
    return grid.best_estimator_

# 让结果在notebook中显示
%matplotlib inline

# Load the Beijing housing dataset
# 载入北京房屋的数据集
data = pd.read_csv('bj_housing.csv')
prices = data['Value']
features = data.drop('Value', axis=1)
print "Beijing housing dataset has {} data points with {} variables each.".format

# 观察原始数据, 决定是否进行预处理
value = data['Value']
for item in ['Area', 'Room', 'Living', 'Year', 'School', 'Floor']:
    idata = data[item]
    matplotlib.pyplot.scatter(idata, value)
    matplotlib.pyplot.show()

# 计算价值的最小值
minimum_price = np.min(prices)
# 计算价值的最大值
maximum_price = np.max(prices)
```

```

#计算价值的平均值
mean_price = np.mean(prices)
#计算价值的中值
median_price = np.median(prices)
#计算价值的标准差
std_price = np.std(prices)
#输出计算的结果
print "Statistics for Beijing housing dataset:\n"
print "Minimum price: ${:,.2f}".format(minimum_price)
print "Maximum price: ${:,.2f}".format(maximum_price)
print "Mean price: ${:,.2f}".format(mean_price)
print "Median price ${:,.2f}".format(median_price)
print "Standard deviation of prices: ${:,.2f}".format(std_price)

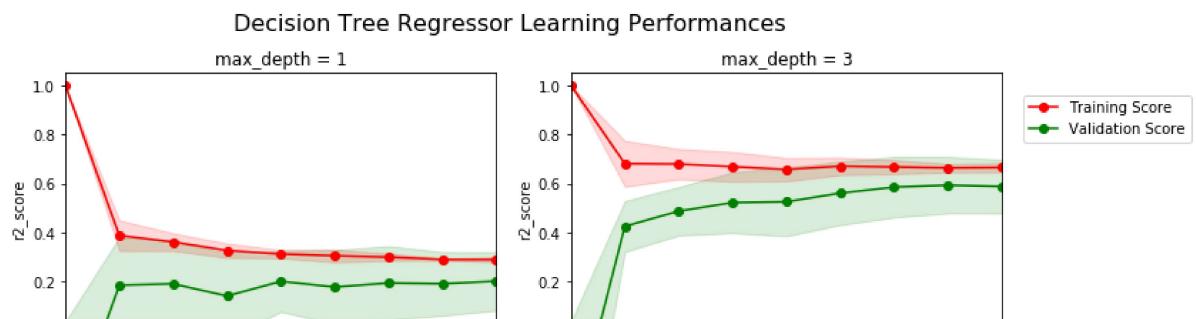
# Train test split
X_train, X_test, y_train, y_test = train_test_split(features, prices, test_size=0)
print("Train test split success!")

# 根据不同的训练集大小, 和最大深度, 生成学习曲线
vs.ModelLearning(X_train, y_train)
# 根据不同的最大深度参数, 生成复杂度曲线
vs.ModelComplexity(X_train, y_train)

# 基于训练数据, 获得最优模型
optimal_reg = fit_model(X_train, y_train)
# 输出最优模型的 'max_depth' 参数
print "Parameter 'max_depth' is {} for the optimal model.".format(optimal_reg.get

# 预测
predicted_price = optimal_reg.predict(X_test)
# 计算R^2的值
r2 = performance_metric(predicted_price, y_test)
print "Optimal model has R^2 score {:.2f} on test data".format(r2)

```



```
In [58]: # Load the Beijing housing dataset
# 载入北京房屋的数据集
data = pd.read_csv('bj_housing2.csv')
prices = data['Value']
features = data.drop('Value', axis=1)
print "Beijing housing dataset has {} data points with {} variables each.".format(len(data), len(features.columns))

# 观察原始数据，决定是否进行预处理
value = data['Value']
for item in ['Area', 'Room', 'Living', 'Year', 'School', 'Floor']:
    idata = data[item]
    matplotlib.pyplot.scatter(idata, value)
    matplotlib.pyplot.show()

# 计算价值的最小值
minimum_price = np.min(prices)
# 计算价值的最大值
maximum_price = np.max(prices)
# 计算价值的平均值
mean_price = np.mean(prices)
# 计算价值的中值
median_price = np.median(prices)
# 计算价值的标准差
std_price = np.std(prices)
# 输出计算的结果
print "Statistics for Beijing housing dataset:\n"
print "Minimum price: ${:,.2f}".format(minimum_price)
print "Maximum price: ${:,.2f}".format(maximum_price)
print "Mean price: ${:,.2f}".format(mean_price)
print "Median price ${:,.2f}".format(median_price)
print "Standard deviation of prices: ${:,.2f}".format(std_price)

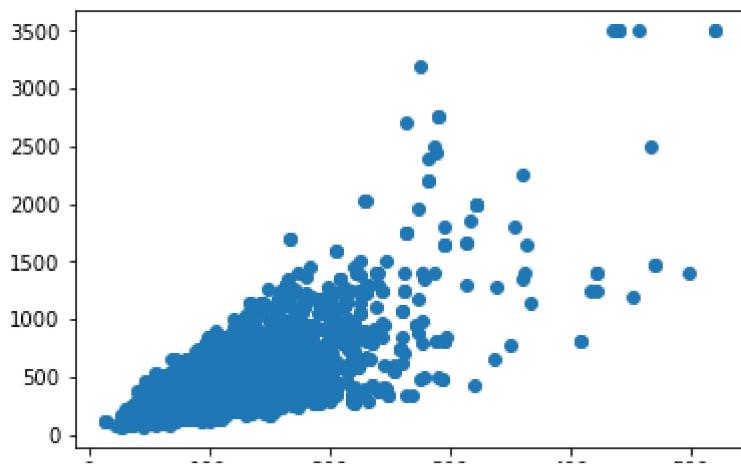
# Train test split
X_train, X_test, y_train, y_test = train_test_split(features, prices, test_size=0)
print("Train test split success!")

# 根据不同的训练集大小，和最大深度，生成学习曲线
vs.ModelLearning(X_train, y_train)
# 根据不同的最大深度参数，生成复杂度曲线
vs.ModelComplexity(X_train, y_train)

# 基于训练数据，获得最优模型
optimal_reg = fit_model(X_train, y_train)
# 输出最优模型的 'max_depth' 参数
print "Parameter 'max_depth' is {} for the optimal model.".format(optimal_reg.get_params()['max_depth'])

# 预测
predicted_price = optimal_reg.predict(X_test)
# 计算R^2的值
r2 = performance_metric(predicted_price, y_test)
print "Optimal model has R^2 score {:.2f} on test data".format(r2)
```

Beijing housing dataset has 9988 data points with 7 variables each.



问题14 - 北京房价预测

你成功的用新的数据集构建了模型了吗？他能对测试数据进行验证吗？它的表现是否符合你的预期？交叉验证是否有助于提升你模型的表现？

提示：如果你是从零开始构建机器学习的代码会让你一时觉得无从下手。这时不要着急，你要做的只是查看之前写的代码，把每一行都看明白，然后逐步构建你的模型。当中遇到什么问题也可以在我们论坛寻找答案。也许你会发现你所构建的模型的表现并没有达到你的预期，这说明机器学习并非是一项简单的任务，构建一个表现良好的模型需要长时间的研究和测试。这也是我们接下来的课程中会逐渐学到的。

问题14 - 回答

通过以上实践，我成功地用新的数据集构建了模型；能够对测试数据进行验证；表现基本符合预期；通过使用交叉验证，有助于提升你模型的表现。

bj_housing.csv: 当最大深度为8时得到最优模型。 Parameter 'max_depth' is 8 for the optimal model. Optimal model has R^2 score 0.36 on test data

发现R²较低，然后再去掉bj_housing.csv中每个属性的一些异常值得到bj_housing2.csv，重复加载训练交叉验证过程。

bj_housing2.csv: 当最大深度为7时得到最优模型。 Parameter 'max_depth' is 7 for the optimal model. Optimal model has R^2 score 0.46 on test data

R²提升显著。

In []: