【华子】机器学习006-用决策树回归器构建房价评估模型 -

(本文所使用的Python库和版本号: Python 3.5, Numpy 1.14, scikit-learn 0.19, matplotlib 2.2)

1）Graphviz 0.8.4 (安装python模块，安装代码: pip install graphviz)

2）安装软件：下载地址https://graphviz.gitlab.io/_pages/Download/windows/graphviz-2.38.msi

3）配置环境变量：D:\Program Files (x86)\Graphviz2.38\bin

最近几十年，房价一直是中国老百姓心中永远的痛，有人说，中国房价就像女人的无肩带文胸，一半人在疑惑：是什么支撑了它?另一半人在等待：什么时候掉下去？ 而女人，永不可能让它掉下来。就算快掉下来了，提一提还是又上去了.....

虽然我们不能预测中国房价什么时候崩盘，但是却可以用机器学习来构建房价预测模型，下面我们使用波士顿房价数据集来建立一个房价评估模型，通过房子所在的地理位置，人口居住特性等因素来综合评估房价。

1. 分析数据集

本次模型所使用的数据集来自于波士顿房价数据集官方网站，地址为：（http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html）。

这个数据集比较小，只有506个样本，每个样本有13个features, 1个label，关于这14个属性的说明如下图所示：

There are **14** attributes in each case of the dataset. They are:

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town
12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population
14. MEDV - Median value of owner-occupied homes in $1000's

所以我们需要构建模型，使用前面的13个特征来评估最后的MEDV，即使用13个特征向量来计算房价，这是典型的多元回归问题。虽然可以从官网上直接下载这个数据集，但是sklearn中已经集成了该数据集，我们可以直接调用。如下代码

```
# 分析数据集
from sklearn import datasets # sklearn自带的datasets中就有Boston房价数据集
housing_data=datasets.load_boston()
dataset_X=housing_data.data # 获取影响房价的特征向量，作为feaure X
dataset_y=housing_data.target # 获取对应的房价，作为label y
# print(dataset_X.shape) # (506, 13) # 一共有506个样本，每个样本有13个features
# print(dataset_y.shape) # (506,)
# print(dataset_X[:5,:]) # 打印看看features的数值类型和大小，貌似已经normalize.

# 上面的数据集划分也可以采用下面的方法：
from sklearn.model_selection import train_test_split
train_X,test_X,train_y,test_y=train_test_split(dataset_X,dataset_y,test_size=0.2,random_state=37
)
```

上述代码首先调用sklearn.load_boston()来获取波士顿房价数据集，然后将该数据集划分为两部分，其中train set占据80%（即404个样本），test set占据20%（102个样本）。在查看数据集中前面五行的结果时，发现整个数据集已经Normalize，故而此处我们没有必要进行归一化。

2. 构建决策树模型

决策树(DecisionTree, DT)是一种常见的用于分类和回归的非参数监督学习方法，目标是创建一个模型，通过从数据特性中推导出简单的决策规则来预测目标变量的值。

决策树模型的优点在于：1，简单容易理解，数据结构可以可视化表达。2，需要很少的数据准备，其他技术通常需要数据标准化，需要创建虚拟变量，并删除空白值。3，能够处理多输出问题。

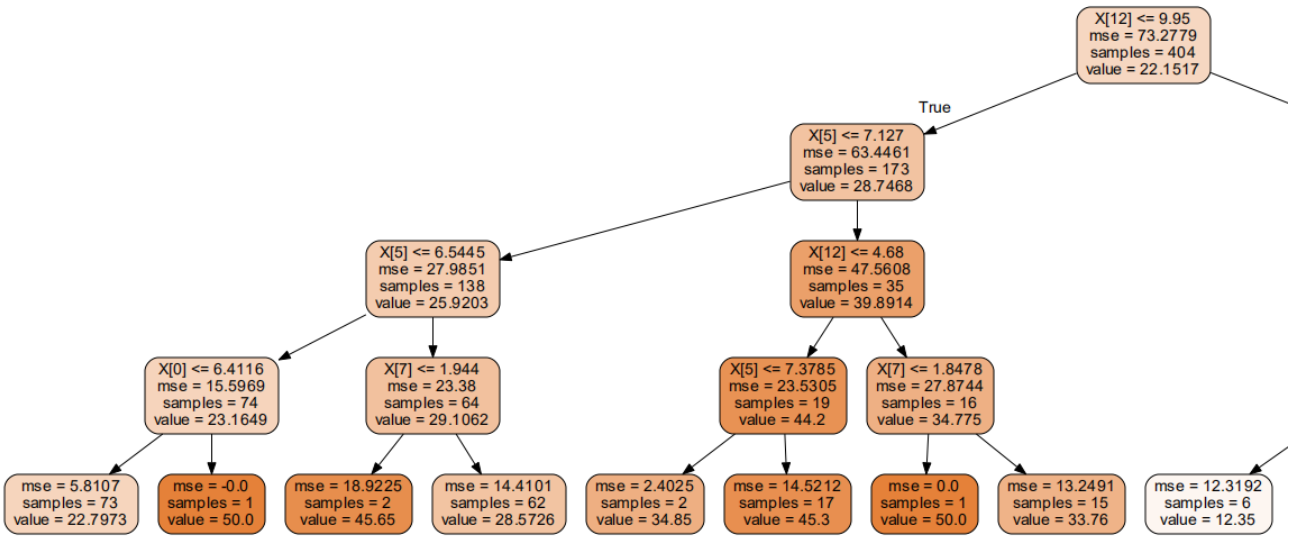
决策树模型的缺点在于：1，决策树学习可能会生成过于复杂的数结构，不能代表普遍的规则，即模型容易过拟合，修剪机制，设置叶子节点所需的最小样本数目或设置树的最大深度是避免决策树过拟合的必要条件。2，决策树可能不稳定，因为数据中的小变化可能导致生成完全不同的树。这个问题可以通过在一个集合中使用多个决策树来减轻。3，实际的决策树学习算法是基于启发式算法的，例如在每个节点上进行局部最优决策的贪婪算法。这种算法不能保证返回全局最优决策树。通过在集合学习中训练多个树，可以减少这种情况，在这里，特征和样本是随机抽取的。

```
# 构建决策树回归模型
from sklearn.tree import DecisionTreeRegressor
decision_regressor=DecisionTreeRegressor(max_depth=4) # 最大深度确定为4
decision_regressor.fit(train_X,train_y) # 对决策树回归模型进行训练

# 使用测试集来评价该决策树回归模型
predict_test_y=decision_regressor.predict(test_X)

# 决策树可视化（最好封装成def）
import graphviz
from sklearn import tree
import os
os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin'
# out_file：输出文件,filled：填充颜色，rounded:圆角
dot_data = tree.export_graphviz(decision_regressor, out_file=None,filled=True, rounded=True)
graph = graphviz.Source(dot_data)
graph.render("tree",view=True)  #在同级目录下生成tree.pdf文件
```

```
#评估结果
import sklearn.metrics as metrics
print('决策树回归模型的评测结果- ---->>>')
print('均方误差MSE：{}'.format(
    round(metrics.mean_squared_error(predict_test_y,test_y),2)))
print('解释方差分：{}'.format(
    round(metrics.explained_variance_score(predict_test_y,test_y),2)))
```

----------输--------出----------

决策树回归模型的评测结果----->>> 均方误差MSE：13.33 解释方差分：0.81



----------完----------

##########小**********结##########

1. 决策树回归模型的构建非常简单，和线性回归器的构建类似，使用DecisionTreeRegressor获取一个回归器对象即可，模型的训练使用fit函数，预测使用predict函数即可。

2. 简单的决策树回归模型得到的MSE比较大，解释方差分结果也不理想，说明这个模型还有很大的优化空间。

##########

3. 决策树模型的优化

一般的，可以先从数据集上优化，但本项目的数据集是非常成熟的案例，进一步优化的空间不大，所以只能从模型上下手。此处我采用两种优化方法，第一种是优化决策树模型中的各种参数，比如优化max_depth，判断是否可以改进MSE，第二种方式是使用AdaBoost算法来增强模型的准确性。

```
# 第一种优化方法：改变max depth来降低MSE,提高解释方差分
for depth in range(2,12):
    decision_regressor_test=DecisionTreeRegressor(max_depth=depth)
    decision_regressor_test.fit(train_X,train_y)
    predict_test_y2=decision_regressor_test.predict(test_X)
    print('depth: {}, MSE: {:.2f}, EVS: {:.2f}'.format(
        str(depth), metrics.mean_squared_error(predict_test_y2,test_y),
        metrics.explained_variance_score(predict_test_y2,test_y)))
```

----------**输**--------**出**----------

depth: 2, MSE: 23.83, EVS: 0.62 depth: 3, MSE: 20.98, EVS: 0.68 depth: 4, MSE: 13.33, EVS: 0.81 depth: 5, MSE: 13.00, EVS: 0.83 depth: 6, MSE: 14.36, EVS: 0.83 depth: 7, MSE: 11.73, EVS: 0.86 depth: 8, MSE: 16.18, EVS: 0.83 depth: 9, MSE: 15.74, EVS: 0.83 depth: 10, MSE: 14.67, EVS: 0.83 depth: 11, MSE: 15.02, EVS: 0.84

----------**完**----------

AdaBoost的做法（以分类原理举例）

- 提高那些被前一轮弱分类器错误分类样本的权值，而降低那些被正确分类样本的权值。
- 加权多数表决的方法，加大分类误差率小的弱分类器的权值，使其在表决中起较大作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。
- 其核心思想是针对同一个训练集训练不同的分类器(弱分类器)，然后把这些弱分类器集合起来，构成一个更强的最终分类器(强分类器)

scikit-learn中Adaboost类库比较直接，就是**AdaBoostClassifier**和**AdaBoostRegressor**两个 。

整个过程如下所示（以分类原理举例）

1. 先通过对 N 个训练样本的学习得到第一个弱分类器；
2. 将分错的样本和其他的新数据一起构成一个新的N 个的训练样本，通过对这个样本的学习得到第二个弱分类器；
3. 将 和 都分错了的样本加上其他的新样本构成另一个新的 N个的训练样本，通过对这个样本的学习得到第三个弱分类器；
4. 如此反复，最终得到经过提升的强分类器。

```
# 第二种优化方法：通过AdaBoost算法来提高模型准确度
from sklearn.ensemble import AdaBoostRegressor
ada_regressor=AdaBoostRegressor(DecisionTreeRegressor(max_depth=7),n_estimators=400)  #最大深度7，基估计其数：400，相当与400棵树
ada_regressor.fit(train_X,train_y)

# 查看使用AdaBoost算法后模型的MSE 和EVS
predict_test_y=ada_regressor.predict(test_X)

import sklearn.metrics as metrics

print('AdaBoost决策树回归模型的评测结果----->>>')
```

```
print('均方误差MSE：{}'.format(
    round(metrics.mean_squared_error(predict_test_y,test_y),2)))
print('解释方差分：{}'.format(
    round(metrics.explained_variance_score(predict_test_y,test_y),2)))
```

----------输--------出----------

AdaBoost决策树回归模型的评测结果----->>> 均方误差MSE：7.87 解释方差分：0.9

----------完----------

##########小**********结##########

1，两种优化方法：第一种通过优化max_depth可以发现，在depth=7时得到的MSE最小，且解释方差分最大，故而认定max_depth=7.

2，第二种通过AdaBoost增强算法来优化模型，得到的MSE（7.87）比depth优化最好的MSE(11.73)要小很多，且解释方差分也有了较大提高，说明优化效果比较好。

3，AdaBoost（Adaptive boosting）算法是一种自适应的利用其他系统来增强模型准确性的算法，将不同版本的算法结果进行组合，用加权汇总的方式获得最终结果。AdaBoost算法在每个阶段获取的信息都会反馈到模型中，这样学习器就可以在后一阶段重点训练难以分类的样本。

4，当然，对于本模型，还可以继续优化，比如优化决策树的结构，优化决策树的其他参数等。

##########

4. 特征的相对重要性

本项目共有13个特征，但这13个特征对于房价的影响肯定是不同的，比如从直观上来看，学区房对房价影响肯定比较大，所以可以计算出各种不同特征对模型的影响，进而在特征比

较复杂的情况下，可以忽略掉影响比较小的特征。

如下两部分代码可以将各种特征的相对重要性绘制到图中。

```
# 计算不同特征的相对重要性
def plot_importances(feature_importances, title, feature_names):
    '''将feature_importance绘制到图表中，便于观察，
    并把重要性大于5的特征打印出来'''
    # 将重要性都归一化为0-100之内
    feature_importances=100.0*(feature_importances/max(feature_importances))

    # ##将得分从高到低排序
    # (np.argsort:将矩阵a按照axis排序(默认升序)，并返回排序后的下标 )
    # (np.flipud:矩阵上下翻转，np.fliplr:矩阵左右反转)
    index_sorted=np.flipud(np.argsort(feature_importances))
    # 让X坐标轴上的标签居中显示
    pos=np.arange(index_sorted.shape[0])+0.5

    # 画条形图
    plt.figure()
    plt.bar(pos,feature_importances[index_sorted],align='center')
```

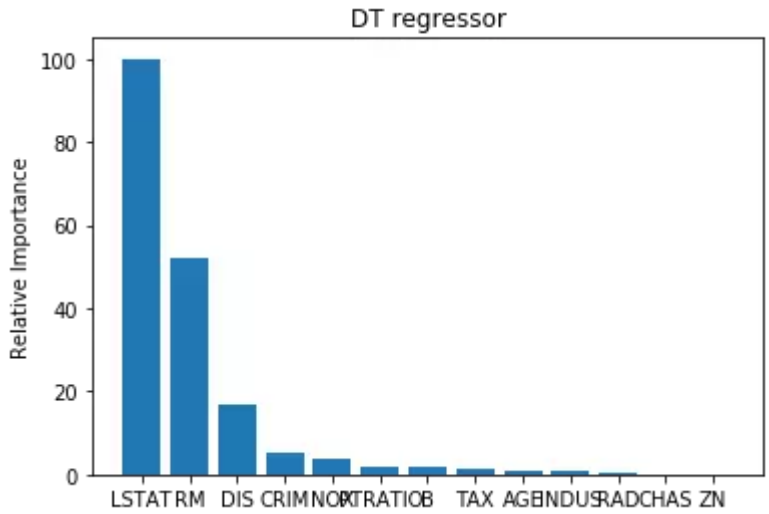
```
    plt.xticks(pos,feature_names[index_sorted])
    plt.ylabel('Relative Importance')
    plt.title(title)

    # 把重要性结果打印出来
    print('{} importance list------>>>>>'.format(title))
    for importance,name in zip(feature_importances[index_sorted],
                               feature_names[index_sorted]):
        if importance>5:
            print('feature:{}, importance: {:.2f}'.format(name,importance))
```

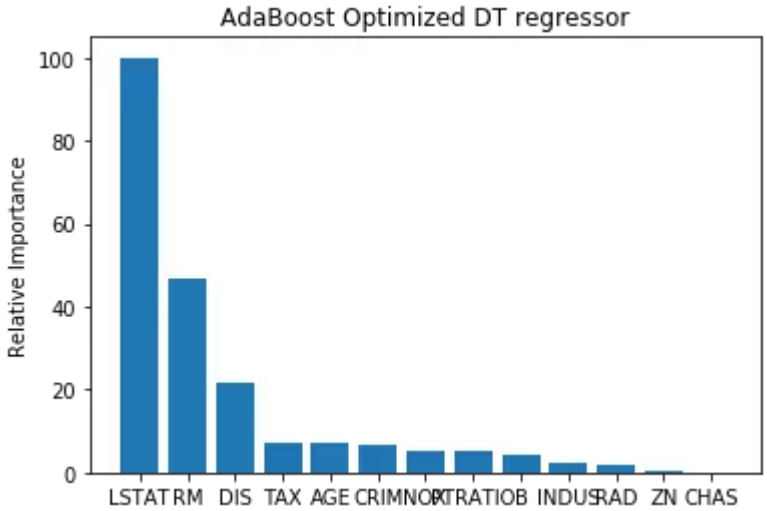
```
decision_regressor7=DecisionTreeRegressor(max_depth=7) # 最大深度确定为7
decision_regressor7.fit(train_X,train_y) # 对决策树回归模型进行训练
plot_importances(decision_regressor7.feature_importances_,
                 'DT regressor',housing_data.feature_names)
plot_importances(ada_regressor.feature_importances_,
                 'AdaBoost Optimized DT regressor',housing_data.feature_names)
```

----------**输--------出**----------

DT regressor importance list------>>>>> feature:LSTAT, importance: 100.00 feature:RM, importance: 52.24 feature:DIS, importance: 15.97 feature:PTRATIO, importance: 5.00 AdaBoost Optimized DT regressor importance list------>>>>> feature:LSTAT, importance: 100.00 feature:RM, importance: 46.72 feature:DIS, importance: 21.80 feature:TAX, importance: 7.34 feature:AGE, importance: 7.29 feature:CRIM, importance: 6.51 feature:NOX, importance: 5.32 feature:PTRATIO, importance: 5.29

----------**完**----------





##########**小**********结**##########

AdaBoost增强算法貌似能够提升其他小特征的相对重要性，而且两种模型中特征重要性排序也不一样。

##########

参考资料:

1, Python机器学习经典实例，Prateek Joshi著，陶俊杰，陈小莉译