决策树、随机森林
GBDT、XGBoost

七月在线 加号

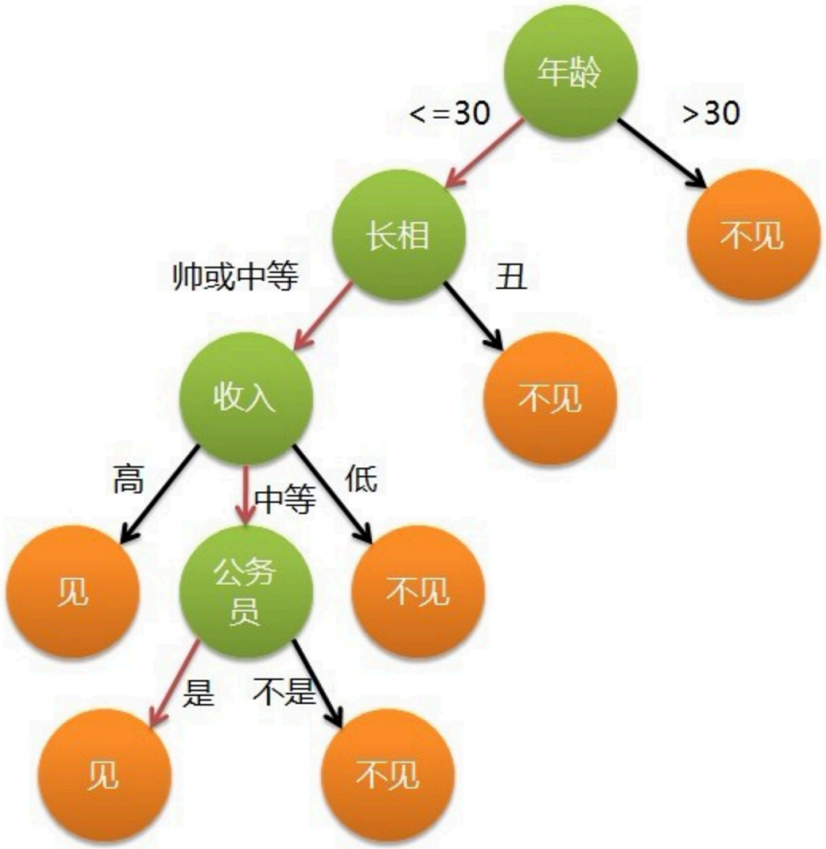
微博：@翻滚吧_加号

目录

- 决策树 Decision Tree
  - 介绍
  - 熵 Entropy
  - 信息增益 Information Gain
  - 常见算法
  - 过度拟合 Overfitting
    - 剪枝 Prune
    - 增益率 GainRatio
  - 更多的DT应用场景类别:
    - 连续函数，多分类，回归
- 决策树的究极进化 Ensemble
  - Bagging
  - Random Forest
  - Boosting
    - GBDT
    - XGBoost

决策树介绍

相亲准则：



案例：预测小明今天出不出门打球

日期	天气	湿度	风级	打球
1	晴	高	弱	否
2	晴	高	强	否
3	阴	高	弱	是
4	雨	高	弱	是
5	雨	正常	弱	是
6	雨	正常	强	否
7	阴	正常	强	是
8	晴	高	弱	否
9	晴	正常	弱	是
10	雨	正常	弱	是
11	晴	正常	强	是
12	阴	高	强	是
13	阴	正常	弱	是
14	雨	高	强	否

决策树介绍

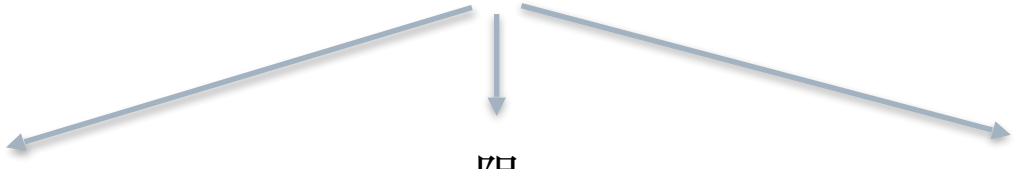
分治思想:

- 把数据集分成两组
- 不同数据点被完美区分（Pure）开了么?
- 不是：重复楼上两步
- 是的：打完收工

案例：预测小明今天出不出门打球

9是 / 5否

天气



晴

1	晴	高	弱	否
2	晴	高	强	否
8	晴	高	弱	否
9	晴	正常	弱	是
11	晴	正常	强	是

2是 / 3否

阴

3	阴	高	弱	是
7	阴	正常	强	是
12	阴	高	强	是
13	阴	正常	弱	是

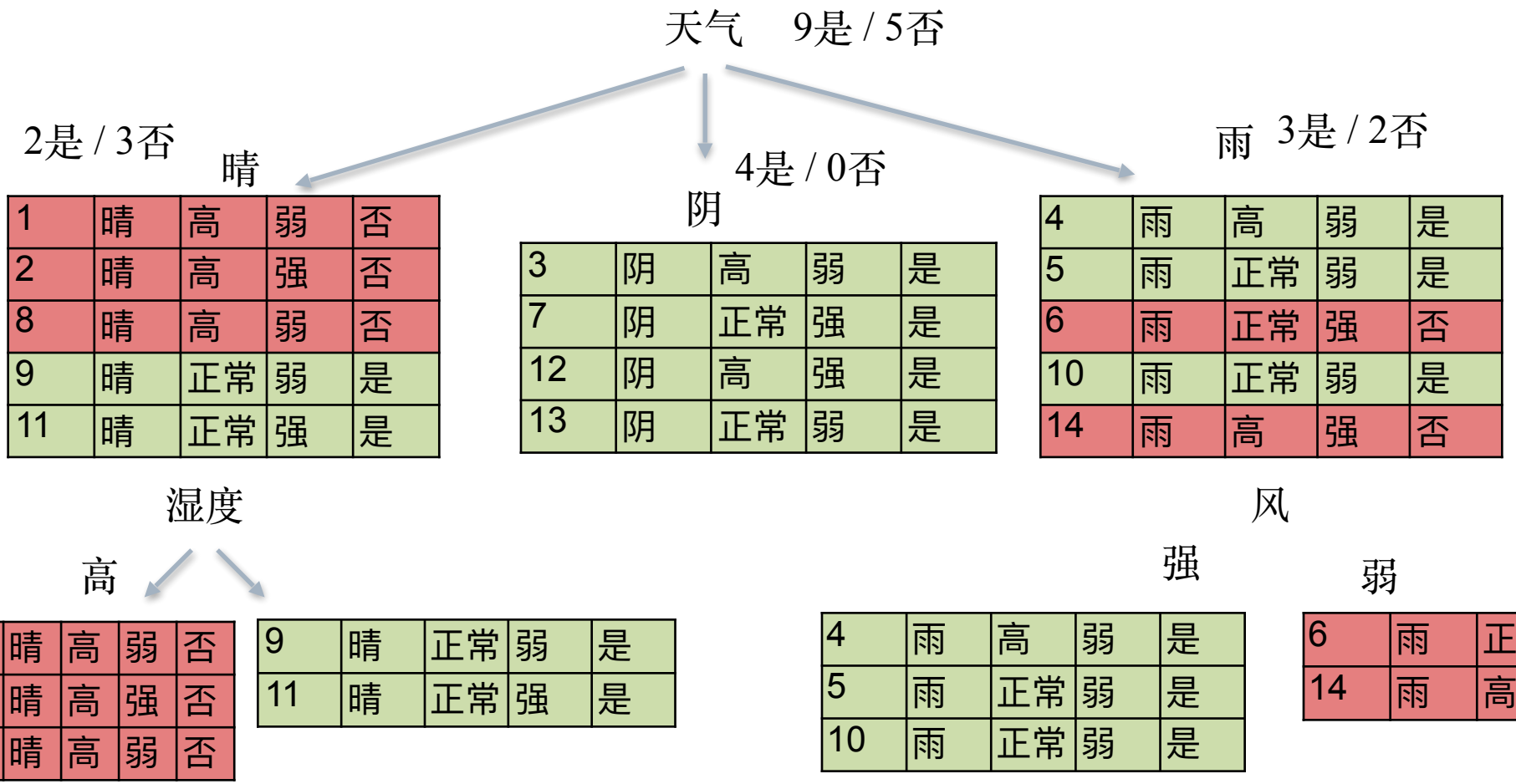
4是 / 0否

雨

4	雨	高	弱	是
5	雨	正常	弱	是
6	雨	正常	强	否
10	雨	正常	弱	是
14	雨	高	强	否

3是 / 2否

案例：预测小明今天出不出门打球



熵 Entropy

怎么知道该选哪个条件（variable）去分开（split）数据?

需要一个衡量Purity（纯洁度）的标准（metrics）：

这个标准需要是symmetric的，就是说 4是/0否 = 0是/4否

熵Entropy：H(S) = - P(是)logP(是) - P(否)logP(否)

不纯洁（impure）：3是/2否
H(S) =

纯洁（pure）：4是/0否
H(S) =

信息增益 Information Gain

$$Gain(S, A) = H(S) - \sum(|S_v|/|S|) * H(S_v)$$

常见算法

ID3算法：

1）对当前样本集合，计算所有属性的信息增益；

2）选择信息增益最大的属性作为测试属性，

把测试属性取值相同的样本划为同一个子样本集；

3）若子样本集的类别属性只含有单个属性，则分支为叶子节点，

判断其属性值并标上相应的符号，然后返回调用处；

否则对子样本集递归调用本算法。

过拟合 Overfitting

ID3 的bug:

1. 永远可以把N个数据分成100%纯洁的N组

2. 如果在『日期』这个条件下做分裂，就GG了

避免过拟合

- 没必要的分裂不要整
- 减枝Prune（根据ValidationSet)

信息增益的缺陷

9是/5否

日期

1	2	3	4	… …	14
0是/1否	0是/1否	1是/0否		… …	0是/1否

信息增益比 Gain Ratio

$$SplitEntropy(S, A) = -\sum \frac{|S_v|}{|S|} log \frac{|S_v|}{|S|}$$

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitEntropy(S, A)}$$

其他常见算法

决策树算法	算法描述
ID3 算法	其核心是在决策树的各级节点上，使用信息增益方法作为属性的选择标准，来帮助确定生成每个节点时所应采用的合适属性。
C4.5 算法	C4.5 决策树算法相对于 ID3 算法的重要改进是使用信息增益率来选择节点属性。C4.5 算法可以克服 ID3 算法存在的不足：ID3 算法只适用于离散的描述属性，而 C4.5 算法既能够处理离散的描述属性，又可以处理连续的描述属性。
CART 算法	CART 决策树是一种十分有效的非参数分类和回归方法，通过构建树、修剪树、评估树来构建一个二叉树。当终节点是连续变量时，该树为回归树；当终节点是分类变量时，该树为分类树

更多的应用场景类别

- Attribute是连续的，而非Categoric
- 多分类 Multi-Class
- 回归Regression

决策树优劣

优势:

- 非黑盒
- 轻松去除无关attribute （Gain = 0）
- Test起来很快 （O(depth)）

劣势:

- 只能线性分割数据
- 贪婪算法（可能找不到最好的树）

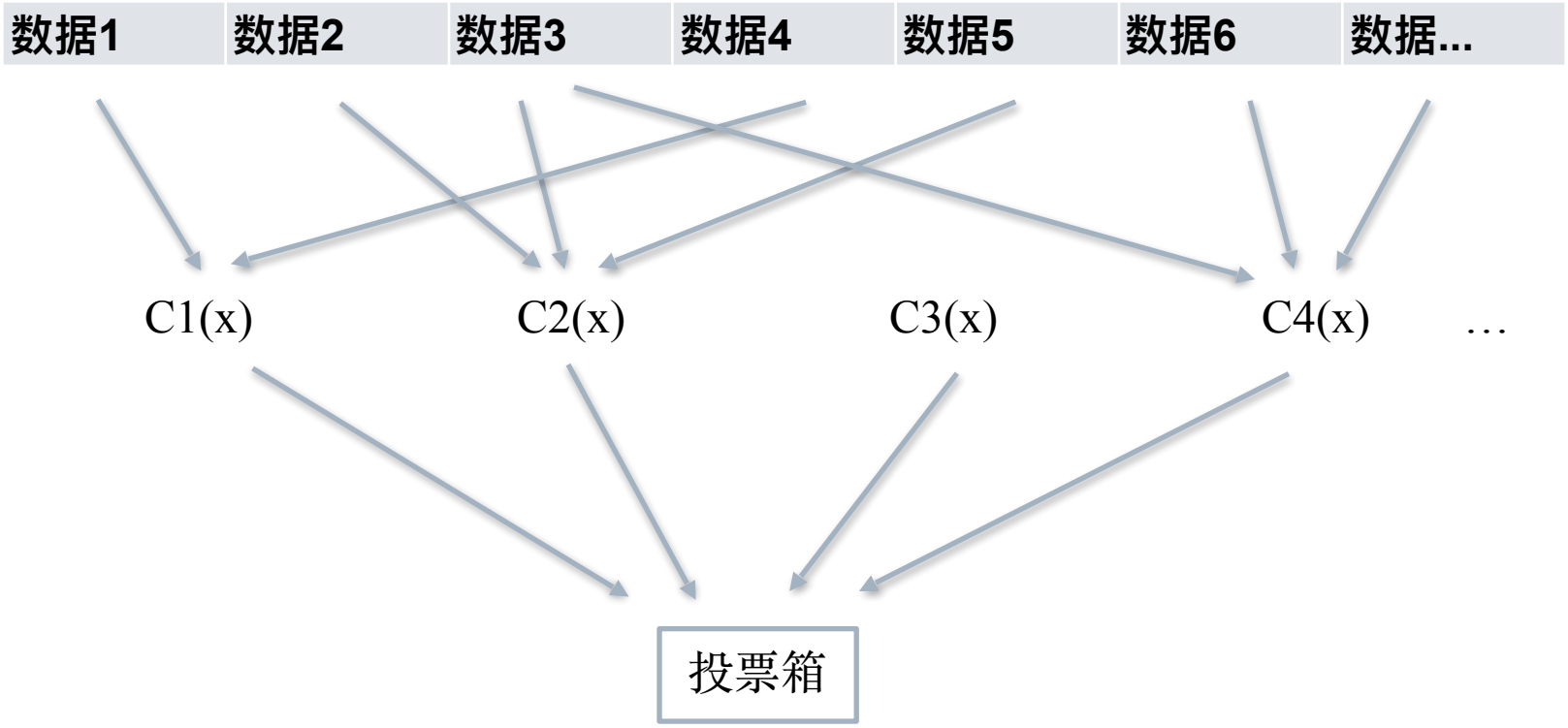
决策树的究极进化 Ensemble

Bagging (Breiman, 1996)
Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.

Random Forest (Freund & Shapire, 1996)
Fancier version of bagging, add one more randomness in variable choosing.

Boosting (Breiman, 1999)
Fit many large or small trees to re-weighted versions of the training data. Classify by weighted majority vote.

Bagging



随机森林 Random Forest

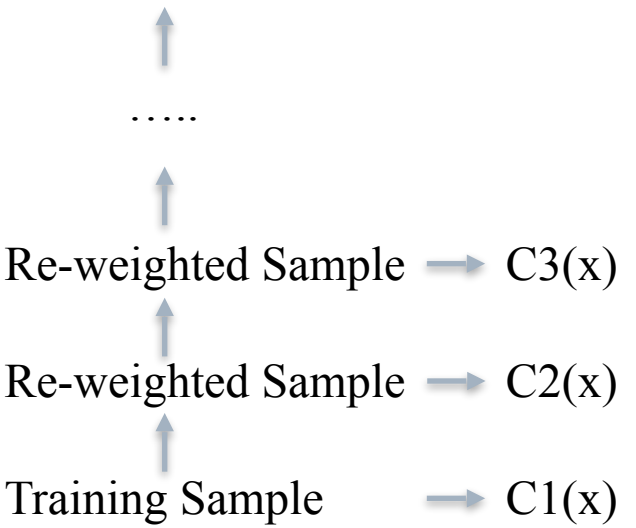
1、从原始训练数据集中，应用bootstrap方法有放回地随机抽取k个新的自助样本集，并由此构建k棵分类回归树，每次未被抽到的样本组成了K个袋外数据（out-of-bag,BBB）。

2、设有n个特征，则在每一棵树的每个节点处随机抽取m个特征，通过计算每个特征蕴含的信息量，特征中选择一个最具有分类能力的特征进行节点分裂。

3、每棵树最大限度地生长， 不做任何剪裁

4、将生成的多棵树组成随机森林， 用随机森林对新的数据进行分类，分类结果按树分类器投票多少而定。

Boosting



1. 先在原数据集中长出一个tree
2. 把前一个tree没能完美分类的数据重新weight
3. 用新的re-weighted tree再训练出一个tree
4. 最终的分类结果由加权投票决定

$$C(x) = sign[\sum a_m C_m(x)]$$

AdaBoost

步骤**1**. 首先，初始化训练数据的权值分布。每一个训练样本最开始时都被赋予相同的权值：1/N

$$D_1 = (w_{11}, w_{12} \cdots w_{1i} \cdots, w_{1N}), \ w_{1i} = \frac{1}{N}, \ i = 1,2,\cdots,N$$

AdaBoost

步骤**2**. 进行多轮迭代，用m = 1,2, ..., M表示迭代的第多少轮

a. 使用具有权值分布Dm的训练数据集学习，得到基本分类器
（选取让误差率最低的阈值来设计基本分类器）：

$$G_m(x): \ \chi \to \{-1,+1\}$$

b. 计算Gm(x)在训练数据集上的分类误差率

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

Note: Gm(x)在训练数据集上的误差率em就是被Gm(x)误分类样本的权值之和。

AdaBoost

c. 计算Gm(x)的系数，am表示Gm(x)在最终分类器中的重要程度
（目的：得到基本分类器在最终分类器中所占的权重）：

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$$

Note: em <= 1/2时，am >= 0，且am随着em的减小而增大，
意味着分类误差率越小的基本分类器在最终分类器中的作用越大。

AdaBoost

d. 更新训练数据集的权值分布（为了得到样本的新的权值分布），用于下一轮迭代

$$D_{m+1}=\left(w_{m+1,1},w_{m+1,2}\cdots w_{m+1,i}\cdots,w_{m+1,N}\right),$$

$$w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp\left(-\alpha_m y_i G_m\left(x_i\right)\right),\ i=1,2,\cdots,N$$

使得被基本分类器Gm(x)误分类样本的权值增大，而被正确分类样本的权值减小。
就这样，通过这样的方式，AdaBoost方法能“重点关注”或“聚焦于”那些较难分的样本上。

其中，Zm是规范化因子，使得Dm+1成为一个概率分布：

$$Z_m=\sum_{i=1}^{N}w_{mi}\exp\left(-\alpha_m y_i G_m\left(x_i\right)\right)$$

AdaBoost

步骤3. 组合各个弱分类器

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$$

从而得到最终分类器，如下：

$$G(x)=sign(f(x))=sign\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)$$

GBDT

Gradient Boosted Decision Tree 的几个特点

a. Adaboost的Regression版本

Adaboost的Error计算

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

GBDT的Error计算

$$e_m = P(G_m(x_i), y_i) = \sum w_{mi} err(G_m(x_i), y_i)$$

GBDT

b. 把残差作为下一轮的学习目标

比如: 用GBDT预测年龄
A的真实年龄是18岁，但第一棵树的预测年龄是12岁，差了6岁，即残差为6岁。
那么在第二棵树里我们把A的年龄设为6岁去学习，如果第二棵树真的能把A分到6岁的叶子节点，
那累加两棵树的结论就是A的真实年龄；如果第二棵树的结论是5岁，则A仍然存在1岁的残差，
第三棵树里A的年龄就变成1岁，继续学。

c. 最终的结果有加权和值得到，不再是简单的多数投票

$$G(x) = \sum a_m G_m(x)$$

XGBoost

本质还是个GBDT，但是是把速度和效率做到了极致，所以叫X (Extreme) GBoosted

a. 使用L1 L2 Regularization 防止Overfitting

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

XGBoost

b. 对代价函数一阶和二阶求导，更快的Converge

c. 树长全后再从底部向上减枝，防止算法贪婪

实战

糖尿病预测

数据集：pima_indians-diabetes.csv
(https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes)

美国亚利桑那州的比马印第安人患糖尿病概率极高。WHO为此调查了21岁以上的女性患者，并记录了以下信息：

1. 怀孕了几次
2. 血糖
3. 血压
4. 皮脂厚度
5. 胰岛素
6. 体质指数
7. 糖尿病血统
8. 年龄
9. label：是否患病

代码

```
# 先导入所有要用的class

import numpy
import xgboost
from sklearn import cross_validation
from sklearn.metrics import accuracy_score

# load数据集
dataset = numpy.loadtxt('pima-indians-diabetes.csv', delimiter=",")

# 把 X Y 分开
X = dataset[:,0:8]
Y = dataset[:,8]
```

代码

```
# 现在我们分开训练集和测试集
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = cross_validation.train_test_split \
(X, Y, test_size=test_size, random_state=seed)

# 训练模型
model = xgboost.XGBClassifier()
# 这里参数的设置可以见：http://xgboost.readthedocs.io/en/latest/python/
python_api.html#module-xgboost.sklearn
model.fit(X_train, y_train)

# 做预测
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]

# 显示准确率
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

代码

```
# 更多的模型:

import sklearn.ensemble.RandomForestClassifier
import sklearn.ensemble.RandomForestRegressor
import sklearn.ensemble.AdaBoostClassifier
import sklearn.ensemble.AdaBosstRegressor
```

感谢大家！

恳请大家批评指正！