法律声明

- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：http://www.ibeifeng.com/

人工智能之机器学习

决策树、随机森林和提升算法

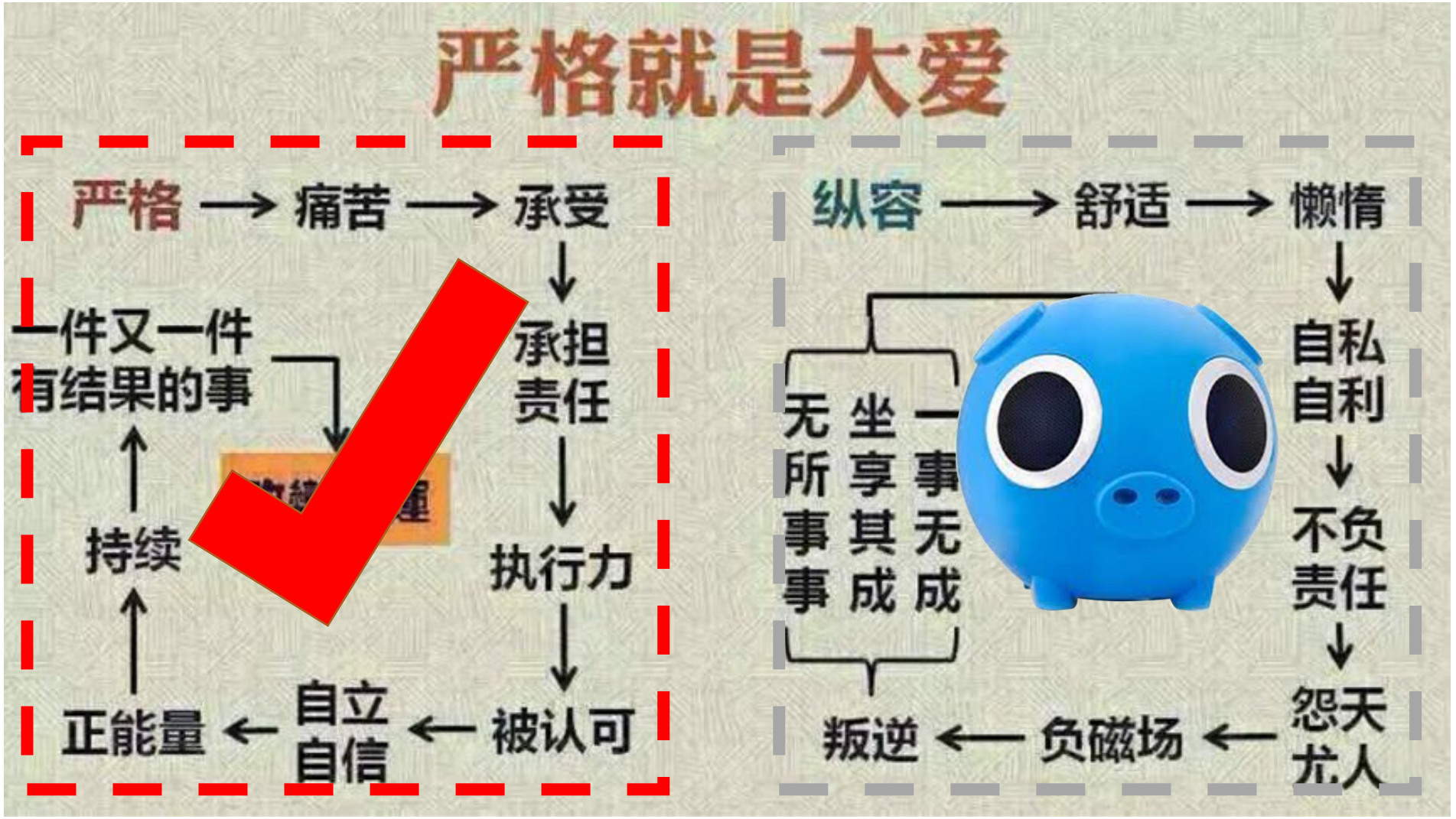
主讲人：June

上海育创网络科技有限公司

课程要求

- 课上课下“九字”真言
  - 认真听，善摘录，勤思考
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不包就业和推荐就业

严格是大爱


严格就是大爱
严格 → 痛苦 → 承受
承担责任
执行力
被认可
自立自信
正能量
持续
一件又一件有结果的事
纵容 → 舒适 → 懒惰
自私自利
不负责任
怨天尤人
负磁场
叛逆
一事无成
坐享其成
无所事事

寄语

做别人不愿做的事，

做别人不敢做的事，

做别人做不到的事。

课程内容

- 信息熵
- 决策树
- 决策树优化
- 剪枝
- 随机森林
- 提升算法
- GBDT(迭代决策树)
- Adaboost

信息熵

如果待分类的事务可能划分在多个分类中，则符号 x_i 的信息定义为：

$$l(x_i) = -\log_2 p(x_i)$$

熵定义为信息的期望值 $$H = -\sum_{i-1}^{n} p(x_i) \log_2 p(x_i)$$

条件熵：

$$H(Y \mid X)$$

$$H(X,Y) - H(X)$$

(X,Y) 发生所包含的熵，减去X单独发生包含的熵，在X发生的前提下，Y发生“新”带来的熵，该式子定义为X发生前提下，Y的熵

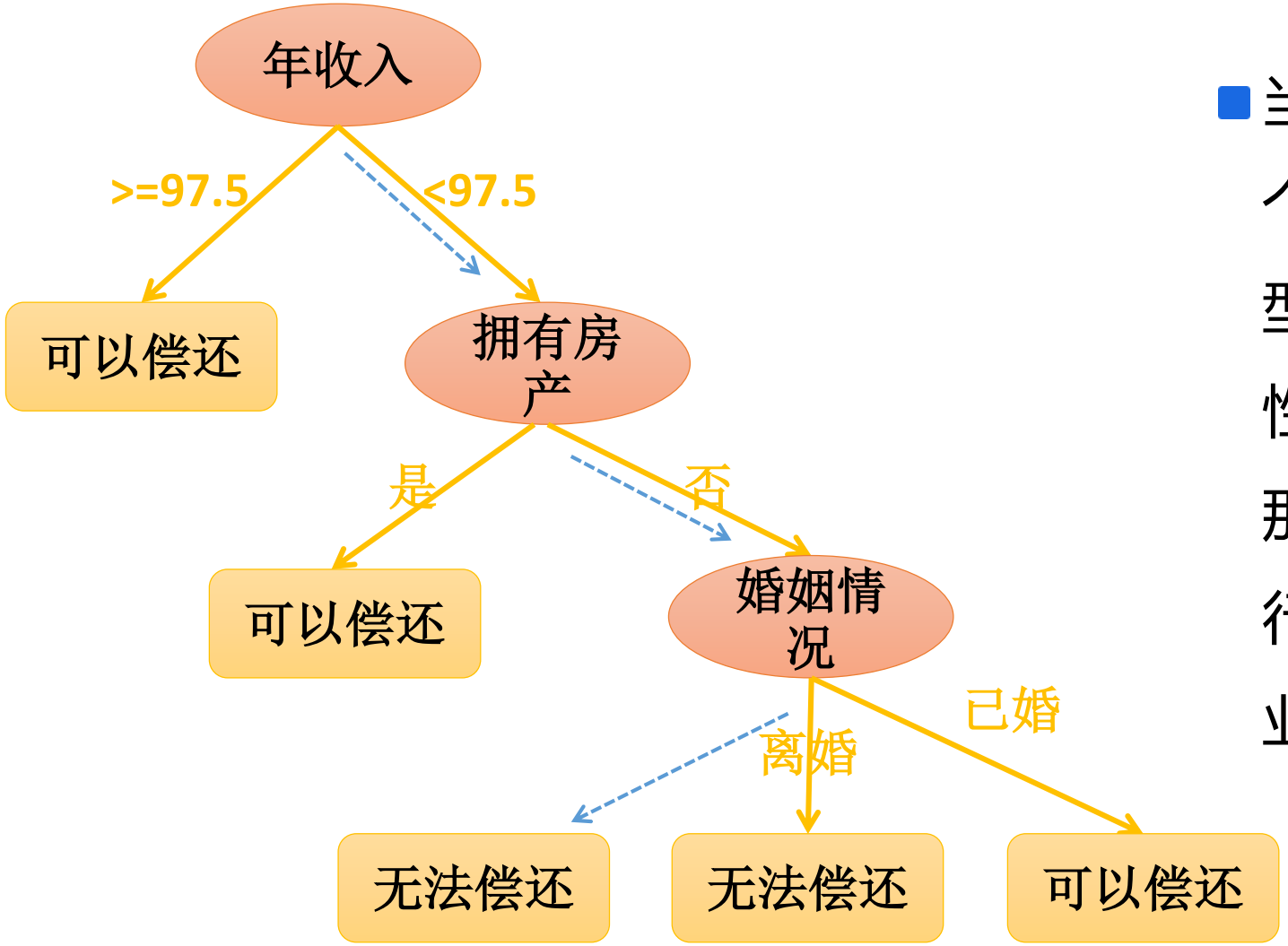
什么是决策树

- 决策树(Decision Tree)是在已知各种情况发生概率的基础上，通过构建决策树来进行分析的一种方式，是一种直观应用概率分析的一种图解法；决策树是一种预测模型，代表的是对象属性与对象值之间的映射关系；决策树是一种树形结构，其中每个内部节点表示一个属性的测试，每个分支表示表示一个测试输出，每个叶节点代表一种类别；决策树是一种非常常用的有监督的分类算法。
- 决策树的决策过程就是从根节点开始，测试待分类项中对应的特征属性，并按照其值选择输出分支，直到叶子节点，将叶子节点的存放的类别作为决策结果。
- 决策树分为两大类：分类树和回归树，前者用于分类标签值，后者用于预测连续值，常用算法有ID3、C4.5、CART等

决策树直观理解

ID	拥有房产(是/否)	婚姻状态(单身\已婚\离婚)	年收入(单位:千元)	无法偿还债务(是/否)
1	是	单身	125	否
2	否	已婚	100	否
3	否	单身	100	否
4	是	已婚	110	否
5	是	离婚	60	否
6	否	离婚	95	是
7	否	单身	85	是
8	否	已婚	75	否
9	否	单身	90	是
10	是	离婚	220	否

决策树直观理解



- 当构建好一个决策树后，新来一个用户后，可以根据决策好的模型直接进行判断，比如新用户特性为：无房产、单身、年收入55K，那么根据判断得出该用户无法进行债务偿还。这种决策对于借贷业务有比较好的指导意义。

决策树构建过程

- 决策树算法的重点就是决策树的构造；决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构(树结构)；构建决策树的关键步骤就是分裂属性，分裂属性是指在某个节点按照某一类特征属性的不同划分构建不同的分支，其目标就是让各个分裂子集尽可能的”纯”(让一个分裂子类中待分类的项尽可能的属于同一个类别)
- 构建步骤如下:
  - 1. 开始，所有记录看做一个节点
  - 2. 遍历每个遍历的每一种分割方式，找到最好的分割点
  - 3. 将数据分割为两个节点部分N1和N2
  - 4. 对N1和N2分别继续执行2-3步，直到每个节点中的项足够”纯”

决策树特征属性类型

- 根据特征属性的类型不同，在构建决策树的时候，采用不同的方式，具体如下：
  - 属性是离散值，而且不要求生成的是二叉决策树，此时一个属性就是一个分支
  - 属性是离散值，而且要求生成的是二叉决策树，此时使用属性划分的子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支
  - 属性是连续值，可以确定一个值作为分裂点split_point，按照>split_point和<=split_point生成两个分支

决策树分割属性选择

- 决策树算法是一种“贪心”算法策略，只考虑在当前数据特征情况下的最好分割方式，不能进行回溯操作。
- 对于整体的数据集而言，按照所有的特征属性进行划分操作，对所有划分操作的结果集的“纯度”进行比较，选择“纯度”越高的特征属性作为当前需要分割的数据集进行分割操作，持续迭代，知道得到最终结果。决策树是通过“纯度”来选择分割特征属性点的。

决策树量化纯度

- 决策树的构建是基于样本概率和纯度进行构建操作的，那么进行判断数据集是否“纯”可以通过三个公式进行判断，分别是Gini系数、熵(Entropy)、错误率，这三个公式值越大，表示数据越“不纯”；越小表示越“纯”；实践证明这三种公式效果差不多，一般情况使用熵公式

$$P(1)=7/10=0.7;\text{可以偿还概率}$$

$$P(2)=3/10=0.3;\text{无法偿还概率}$$

$$Gini=1-\sum_{i=1}^{n}P(i)^2 \qquad H(Entropy)=-\sum_{i=1}^{n}P(i)\log_2(P(i)) \qquad Error=1-\max_{i=1}^{n}\{P(i)\}$$

决策树量化纯度

- 当计算出各个特征属性的量化纯度值后使用**信息增益度**来选择出当前数据集的分割特征属性；如果信息增益度的值越大，表示在该特征属性上会损失的纯度越大 ，那么该属性就越应该在决策树的上层，计算公式为：

$$Gain = \Delta = \mathrm{H}(\mathrm{D}) - \mathrm{H}(D / \mathrm{A})$$

- Gain为A为特征对训练数据集D的信息增益，它为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D /A)之差

信息增益

- 特征A的信息增益，g(D/A)=H(D)-H(D /A)

$$H(D \mid A) = -\sum_{i,k} p(D_k, A_i) \log p(D_k \mid A_i)$$

$$= -\sum_{i,k} p(A_i) p(D_k \mid A_i) \log p(D_k \mid A_i)$$

$$= -\sum_{i=1}^{n} \sum_{k=1}^{K} p(A_i) p(D_k \mid A_i) \log p(D_k \mid A_i)$$

$$= -\sum_{i=1}^{n} p(A_i) \sum_{k=1}^{K} p(D_k \mid A_i) \log p(D_k \mid A_i)$$

$$= -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log \frac{|D_{ik}|}{|D_i|}$$

设特征A有n个不同的取值{a1,a2,……an}，根据特征A的取值将D划分为n个子集 $D_1, D_2 \cdots D_n$ ，$|D_i|$ 为 D_i 的样本个数 $\sum_i |D_i| = |D|$

决策树算法的停止条件

- 决策树构建的过程是一个递归的过程，所以必须给定停止条件，否则过程将不会进行停止，一般情况有两种停止条件：
  - 当每个子节点只有一种类型的时候停止构建
  - 当前节点中记录数小于某个阈值，同时迭代次数达到给定值时，停止构建过程，此时使用max(p(i))作为节点的对应类型
- 方式一可能会使树的节点过多，导致过拟合(Overfiting)等问题；比较常用的方式是使用方式二作为停止条件

决策树算法效果评估

- 决策树的效果评估和一般的分类算法一样，采用混淆矩阵来进行计算准确率、召回率、精确率等指标
- 也可以采用叶子节点的纯度值总和来评估算法的效果，值越小，效果越好

		predicted condition			
	total population	prediction positive	prediction negative	Prevalence = $\frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$	
true condition	condition positive	True Positive (TP)	False Negative (FN) (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection = $\frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$	False Negative Rate (FNR), Miss Rate = $\frac{\Sigma \text{ FN}}{\Sigma \text{ condition positive}}$
	condition negative	False Positive (FP) (Type I error)	True Negative (TN)	False Positive Rate (FPR), Fall-out, Probability of False Alarm = $\frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$	True Negative Rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ TN}}{\Sigma \text{ condition negative}}$
	Accuracy = $\frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision = $\frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) = $\frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic Odds Ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False Discovery Rate (FDR) = $\frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) = $\frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	

$$C(T) = \sum_{t=1}^{leaf} D_t H(t)$$

决策树算法效果评估

- 决策树的损失函数(该值越小，算法效果越好)

$$loss = \sum_{t=1}^{leaf} D_t H(t)$$

决策树直观理解结果计算

ID	拥有房产(是/否)	婚姻状态(单身\已婚\离婚)	年收入(单位:千元)	无法偿还债务(是/否)
1	是	单身	125	否
2	否	已婚	100	否
3	否	单身	100	否
4	是	已婚	110	否
5	是	离婚	60	否
6	否	离婚	95	是
7	否	单身	85	是
8	否	已婚	75	否
9	否	单身	90	是
10	是	离婚	220	否

$$Info(D) = -\sum_{i=1}^{2} P(i)\log_2(P(i)) = -0.7*\log_2(0.7) - 0.3\log_2(0.3) = 0.88$$

$$Info(D_{有房产}) = -4/4*\log_2(4/4) - 0*\log_2(0) = 0 \qquad Info(D_{无房产}) = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$$

$$Gain(房产) = Info(D) - \sum_{j=1}^{2} \frac{N(D_j)}{N(D)} Info(D_j) = 0.88 - 0.4*0 - 0.6*1 = 0.28$$

$$Gain(婚姻) = 0.205 \qquad Gain(收入 = 97.5) = 0.395$$

第一个分割属性为收入

ID3算法

- ID3算法是决策树的一个经典的构造算法，内部使用**信息熵**以及**信息增益**来进行构建；每次迭代选择信息增益最大的特征属性作为分割属性

$$\mathrm{H}(\mathrm{D}) = -\sum_{i=1}^{n} P(i)\log_2(P(i))$$

$$Gain = \Delta = \mathrm{H}(\mathrm{D}) - \mathrm{H}(D \mid \mathrm{A})$$

ID3算法优缺点

- 优点:
  - 决策树构建速度快；实现简单；
- 缺点：
  - 计算依赖于特征数目较多的特征，而属性值最多的属性并不一定最优
  - ID3算法不是递增算法
  - ID3算法是单变量决策树，对于特征属性之间的关系不会考虑
  - 抗噪性差
  - 只适合小规模数据集，需要将数据放到内存中

C4.5算法

- 在ID3算法的基础上，进行算法优化提出的一种算法(C4.5)；现在C4.5已经是特别经典的一种决策树构造算法；使用**消息增益率**来取代ID3算法中的消息增益，在树的构造过程中会进行**剪枝**操作进行优化；能够自动完成对连续属性的离散化处理；C4.5算法在选中分割属性的时候选择信息增益率最大的属性，涉及到的公式为：

$$\mathrm{H}(\mathrm{D}) = -\sum_{i=1}^{n} P(i)\log_2(P(i))$$

$$Gain(A) = \Delta = \mathrm{H}(\mathrm{D}) - \mathrm{H}(D \mid \mathrm{A})$$

$$Gain\_ratio(A) = \frac{Gain(A)}{\mathrm{H}(A)}$$

C4.5算法优缺点

- 优点：
  - 产生的规则易于理解
  - 准确率较高
  - 实现简单
- 缺点：
  - 对数据集需要进行多次顺序扫描和排序，所以效率较低
  - 只适合小规模数据集，需要将数据放到内存中

ID3算法和C4.5算法总结

- ID3和C4.5算法均只适合在小规模数据集上使用
- ID3和C4.5算法都是单变量决策树
- 当属性值取值比较多的时候，最好考虑C4.5算法，ID3得出的效果会比较差
- 决策树分类一般情况只适合小数据量的情况(数据可以放内存)

CART算法

- 使用**基尼系数**作为数据纯度的量化指标来构建的决策树算法就叫做CART(Classification And Regression Tree，分类回归树)算法。CART算法使用**GINI增益率**作为分割属性选择的标准，选择GINI增益率最大的作为当前数据集的分割属性；可用于分类和回归两类问题。

$$Gini = 1 - \sum_{i=1}^{n} P(i)^2$$

$$Gain(A) = \Delta = H(D) - H(D / A)$$

$$Gain\_ratio(A) = \frac{Gain(A)}{H(A)}$$

决策树案例一：鸢尾花数据分类

- 使用决策树算法API对鸢尾花数据进行分类操作，并理解及进行决策树API的相关参数优化
- 数据来源：鸢尾花数据

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Iris Data Set

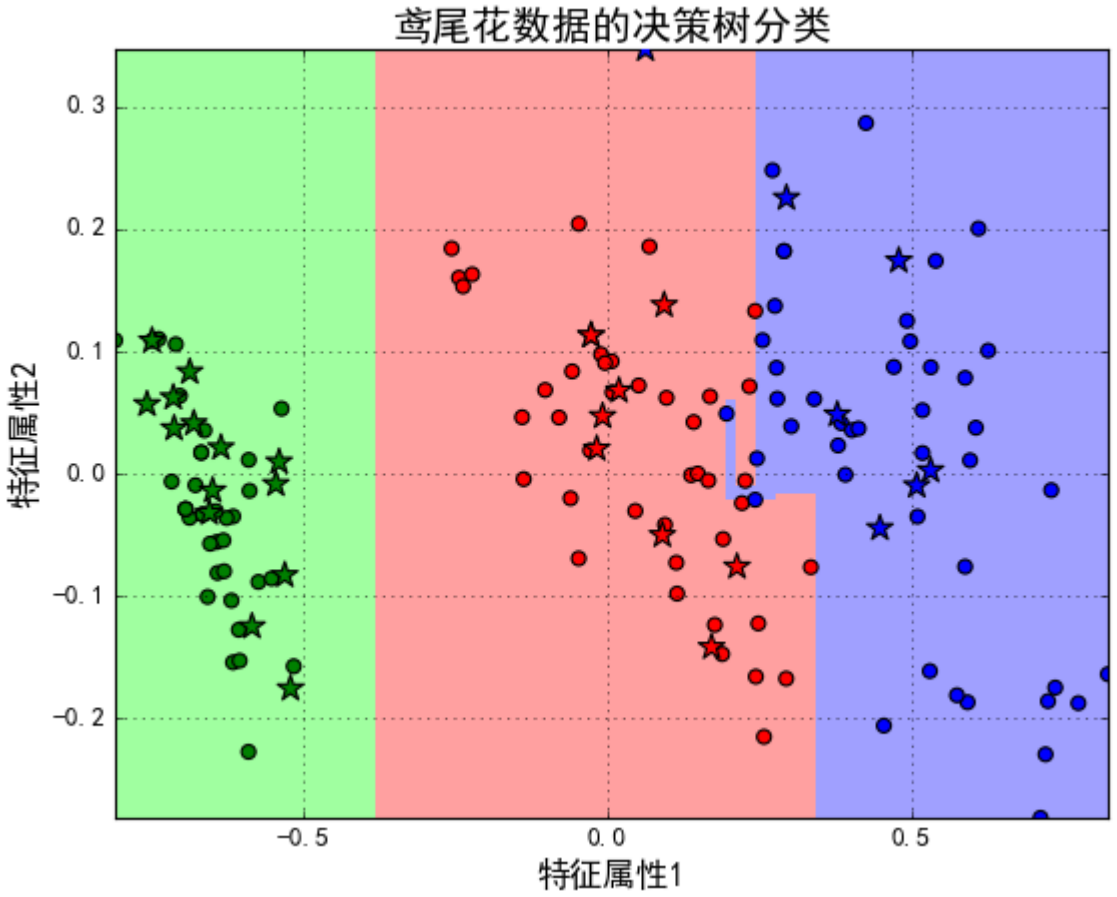
Download: Data Folder, Data Set Description

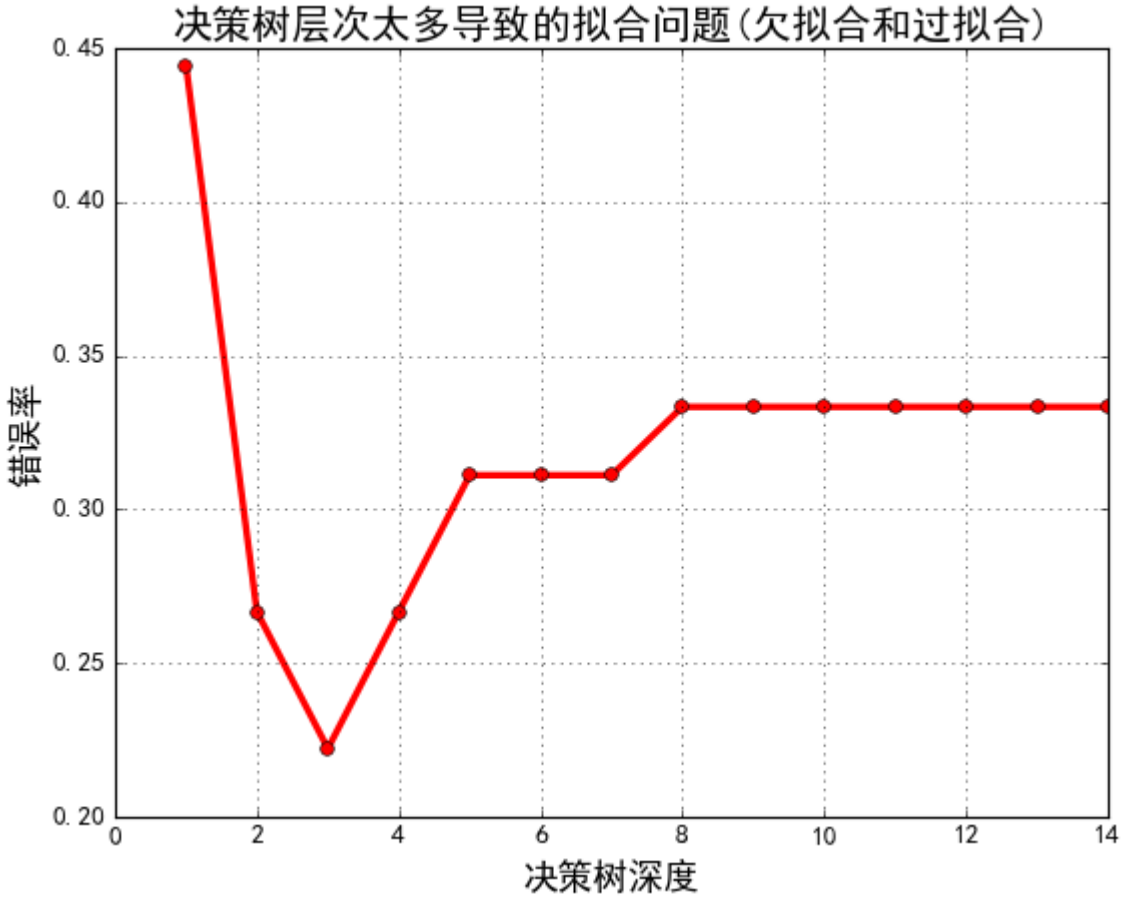
Abstract: Famous database; from Fisher, 1936

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1323697

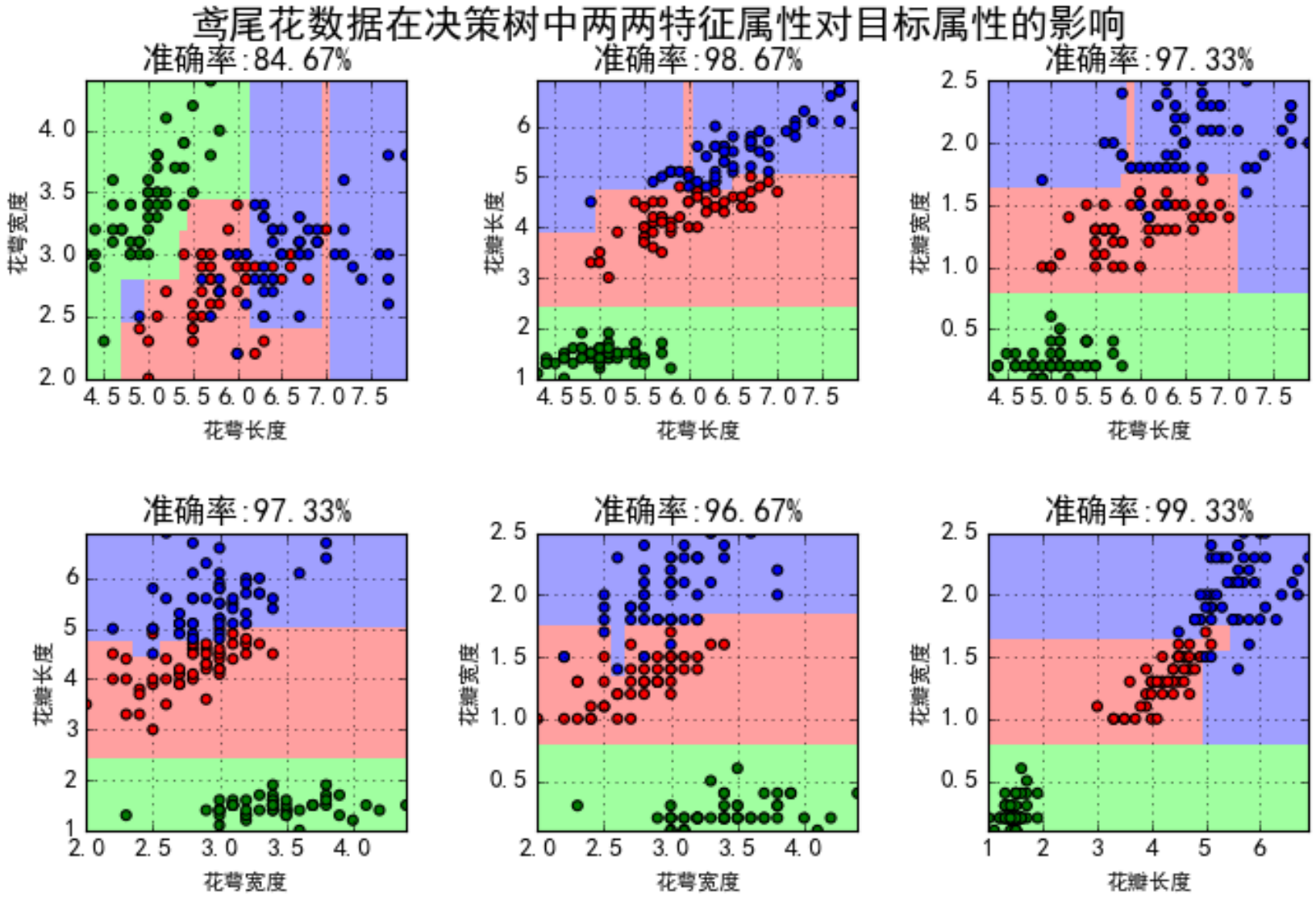
class sklearn.tree.**DecisionTreeClassifier**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, class_weight=None*) ¶ [source]

决策树案例一：鸢尾花数据分类





决策树案例一：鸢尾花数据分类



决策树案例二：波士顿房屋租赁价格预测

- 使用决策树算法API对波士顿房屋租赁数据进行回归操作，预测房屋的价格信息，并理解及进行决策树API的相关参数优化
- 数据来源：波士顿房屋租赁数据

Housing Data Set

Download: Data Folder, Data Set Description

Abstract: Taken from StatLib library

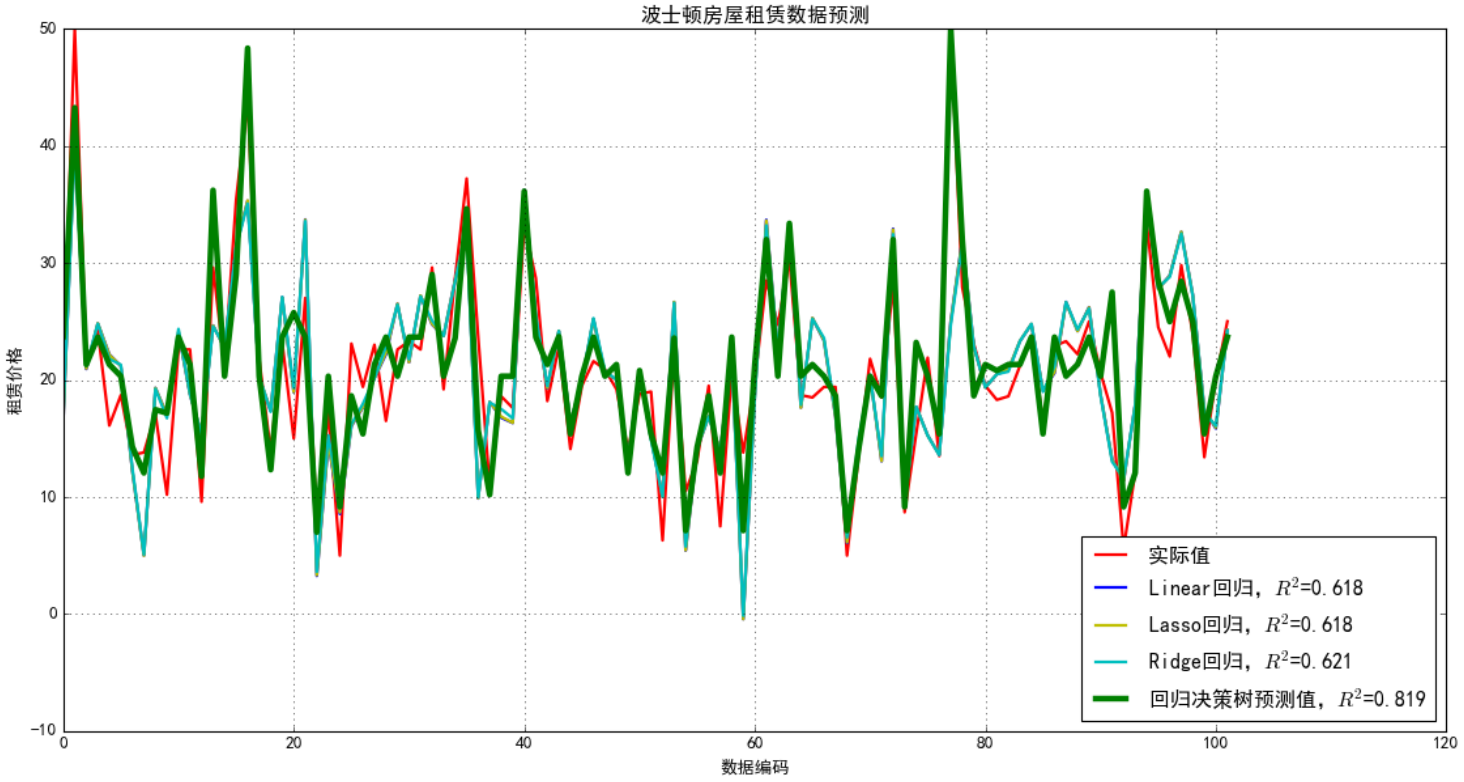
Data Set Characteristics:	Multivariate	**Number of Instances:**	506	**Area:**	N/A
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	14	**Date Donated**	1993-07-07
Associated Tasks:	Regression	**Missing Values?**	No	**Number of Web Hits:**	328263

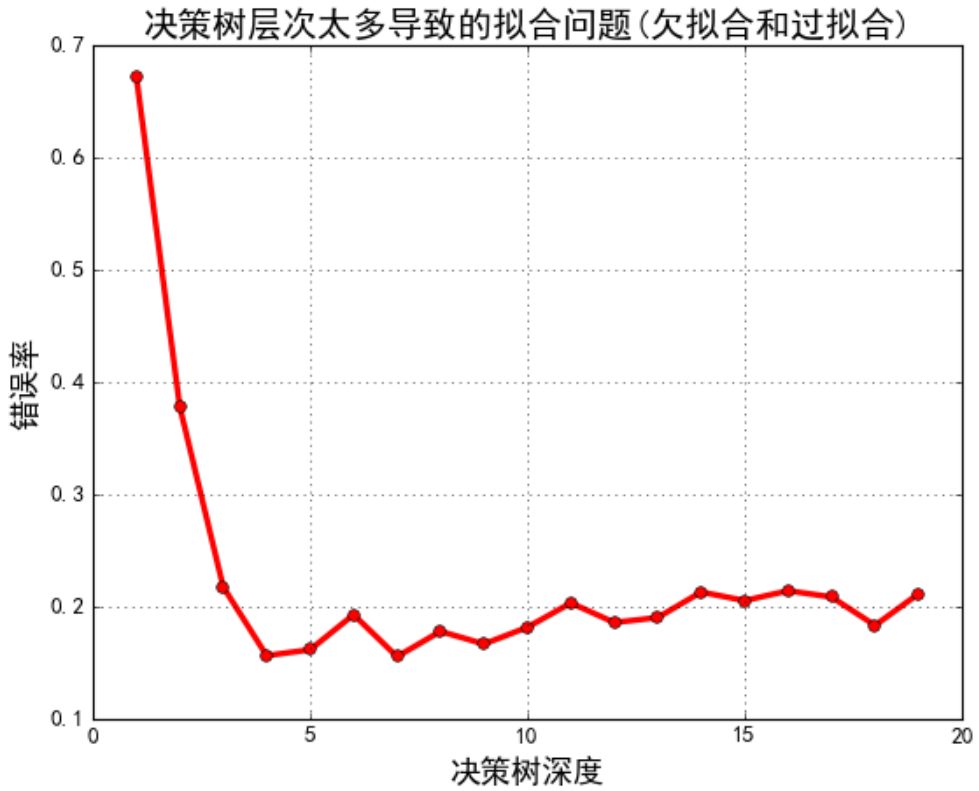
Attribute Information:

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

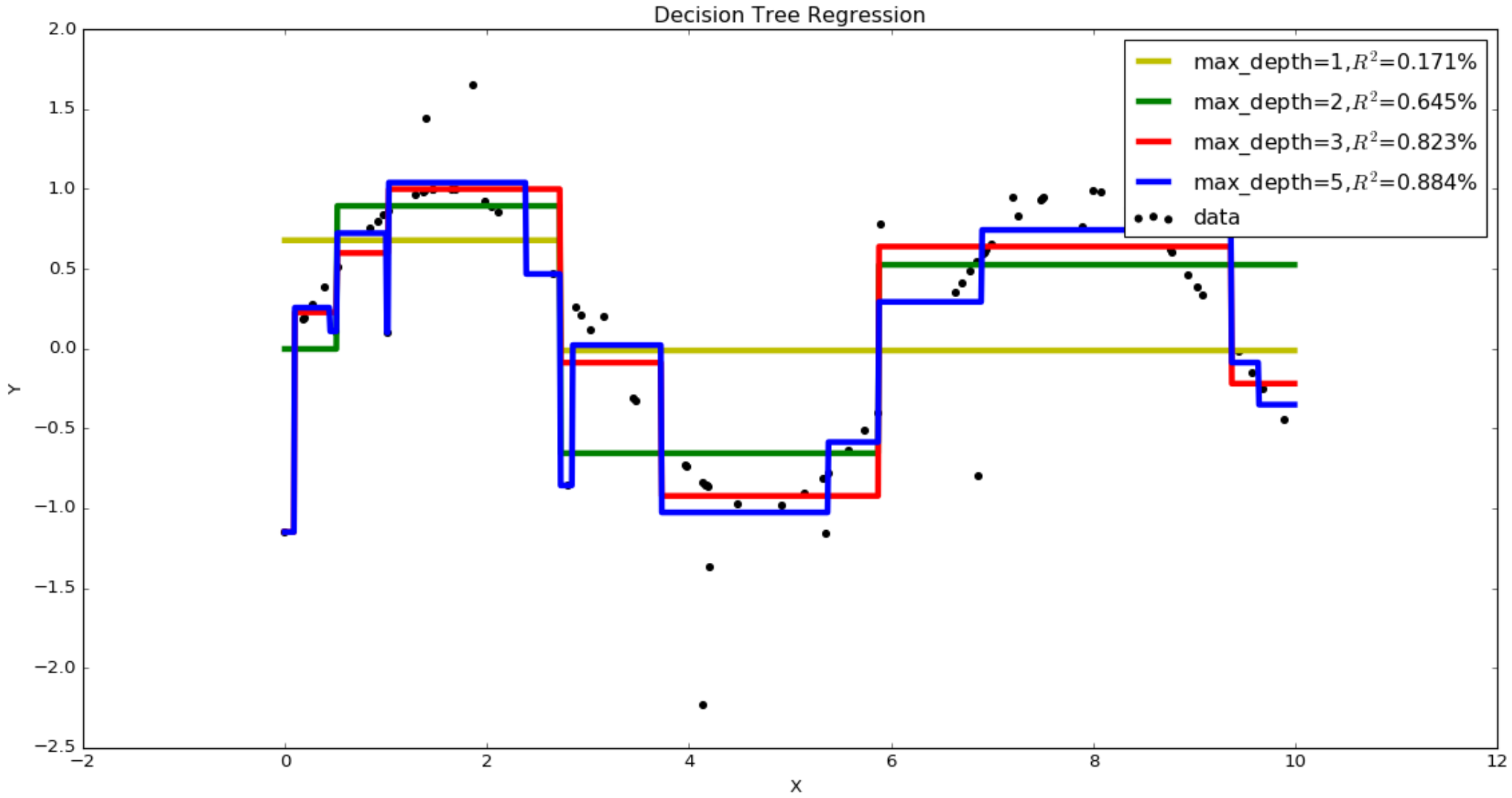
class sklearn.tree.**DecisionTreeRegressor**(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None*) ¶ [source]

决策树案例二：波士顿房屋租赁价格预测





决策树过拟合和欠拟合


Decision Tree Regression
max_depth=1,R^2=0.171%
max_depth=2,R^2=0.645%
max_depth=3,R^2=0.823%
max_depth=5,R^2=0.884%
data
X
Y
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−2
0
2
4
6
8
10
12

决策树优化策略

- 剪枝优化
  - 决策树过渡拟合一般情况是由于节点太多导致的，剪枝优化对决策树的正确率影响是比较大的，也是最常用的一种优化方式。
- K-Fold Cross Validation(K交叉验证)
  - 首先计算出整体的决策树T，叶子点个数为N，记i属于[1,N]。对每个i，使用K交叉验证方法计算决策树，并使决策树的叶子节点数量为i个(剪枝)，计算错误率，得出平均错误率；使用最小错误率的i作为最优决策树的叶子点数量，并对原始决策树T进行裁剪。
- Random Forest
  - 利用训练数据随机产生多个决策树，形成一个森林。然后使用这个森林对数据进行预测，选取最多结果作为预测结果。

决策树的剪枝

- 决策树的剪枝是决策树算法中最基本、最有用的一种优化方案，主要分为两大类：
  - **前置剪枝**：在构建决策树的过程中，提前停止。结果是决策树一般比较小，实践证明这种策略无法得到比较好的结果。
  - **后置剪枝**：在决策树构建好后，然后再开始裁剪，一般使用两种方式：1)用单一叶子节点代替整个子树，叶节点的分类采用子树中最主要的分类；2)将一个子树完全替代另外一棵子树；后置剪枝的主要问题是计算效率问题，存在一定的浪费情况。
- 剪枝总体思路：
  - 由完全树T_0开始，剪枝部分节点得到T_1，在此剪枝得到T_2.....直到仅剩树根的树T_k
  - 在**验证数据集**上对这k+1个树进行评价，选择最优树T_a（损失函数最小的树）

决策树剪枝损失函数

- 原始损失函数

$$loss = \sum_{t=1}^{leaf} \mathrm{D_t H}(t)$$

- 叶节点越多，决策树越复杂，损失越大，添加剪枝系数，修改后的损失函数为：

$$loss_\alpha = loss + \alpha * leaf$$

- 考虑根节点为r的子树，剪枝前后的损失函数分别为loss(r)和loss(R)，当这两者相等的时候，可以求得剪枝系数

$$loss_\alpha(r) = loss(r) + \alpha \qquad loss_r(R) = loss(R) + \alpha * R_{leaf}$$

$$\alpha = \frac{loss(r) - loss(R)}{R_{leaf} - 1}$$

决策树剪枝过程

- 对于给定的决策树T_0
  - 计算所有内部非叶子节点的**剪枝系数**
  - 查找**最小剪枝系数**的节点，将其子节点删除，进行剪枝得到决策树T_k
  - 如果存在多个最小剪枝系数节点，选择包含**数据项最多**的节点进行剪枝操作
  - 对剪枝的决策树T_k重复以上步骤，直到产生的剪枝决策树T_k只有1个节点
  - 得到决策树$T_0T_1T_2....T_k$
  - 使用**验证样本集**选择最优子树T_a
- 使用验证集选择最优子树的标准，可以使用原始损失函数来考虑：

$$loss = \sum_{t=1}^{leaf} D_t H(t)$$

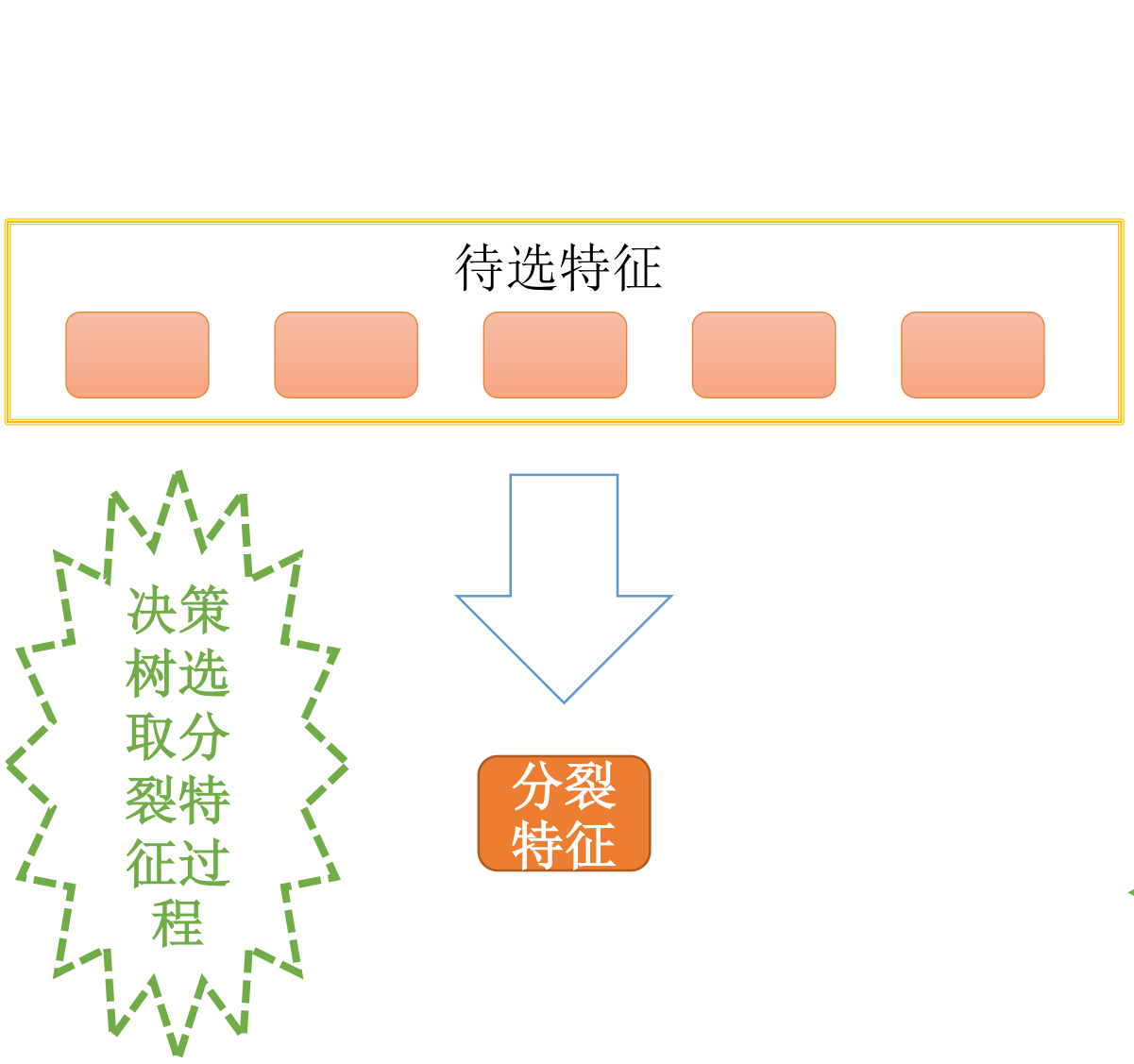
Bagging的策略

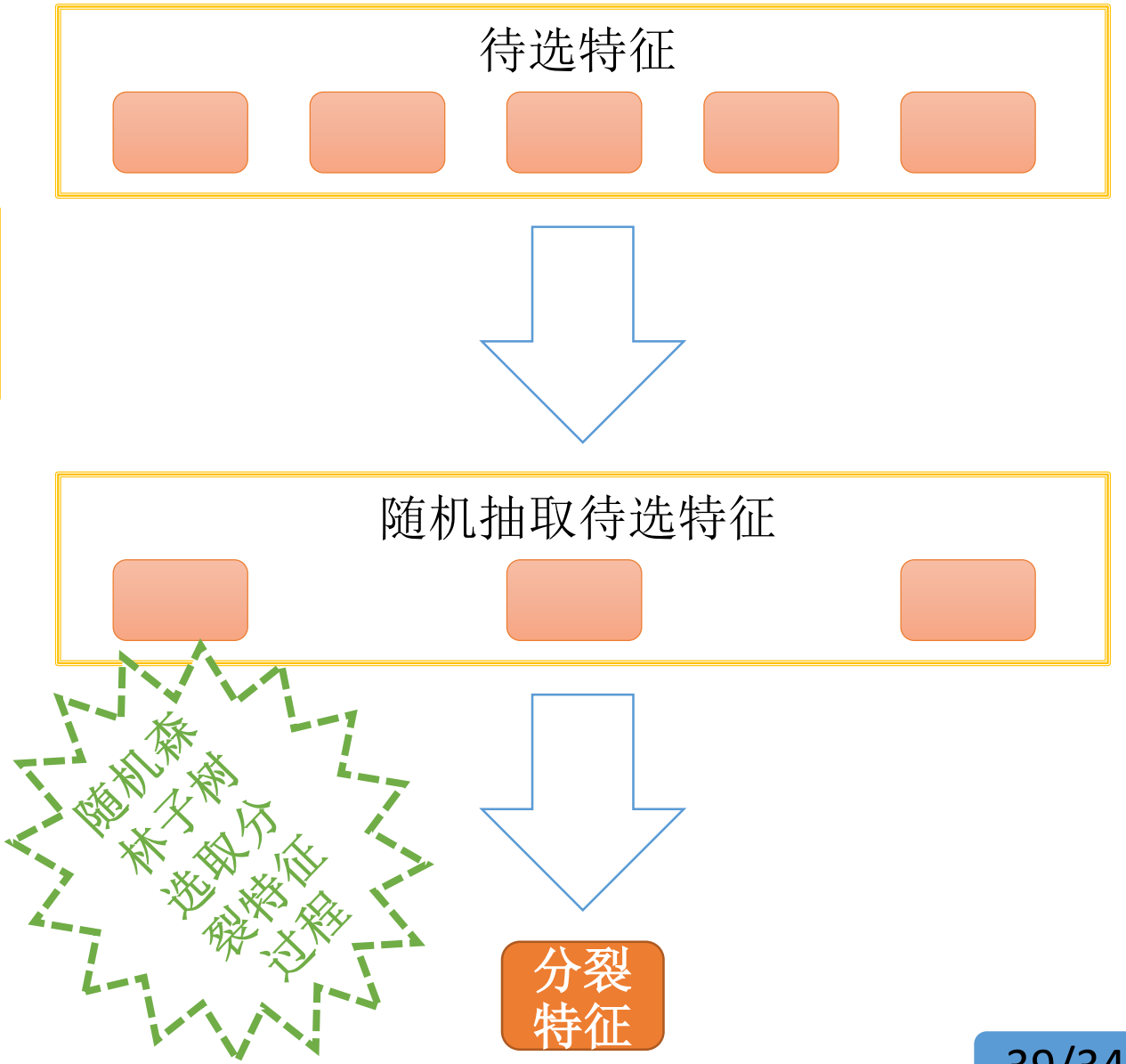
- bootstrap aggregation
- 从样本集中重采样(有可能存在重复)选出n个样本
- 在所有属性上，对这n个样本建立分类器(**ID3**、**C4.5**、**CART**、SVM、Logistic回归等)
- 重复上面两步m次，产生m个分类器
- 将待预测数据放到这m个分类器上，最后根据这m个分类器的投票结果，决定待预测数据属于那一类(即少数服从多数的策略)

随机森林(Random Forest)

- 在Bagging策略的基础上进行修改后的一种算法
  - 从样本集中用Bootstrap采样选出n个样本；
  - 从所有属性中随机选择K个属性，选择出最佳分割属性作为节点创建决策树；
  - 重复以上两步m次，即建立m棵CART决策树；
  - 这m个CART形成随机森林，通过投票表决结果决定数据属于那一类

随机森林(Random Forest)





随机森林算法案例

- 使用随机森林算法API对乳腺癌数据进行分类操作，根据特征属性预测是否会得乳腺癌的四个目标属性的值，并理解随机森林中决策树数量和决策树深度对模型的影响
- 数据来源：乳腺癌数据

(bool) Hinselmann: target variable
(bool) Schiller: target variable
(bool) Cytology: target variable
(bool) Biopsy: target variable

目标属性

Attribute Information:

(int) Age
(int) Number of sexual partners
(int) First sexual intercourse (age)
(int) Num of pregnancies
(bool) Smokes
(bool) Smokes (years)
(bool) Smokes (packs/year)
(bool) Hormonal Contraceptives
(int) Hormonal Contraceptives (years)
(bool) IUD
(int) IUD (years)
(bool) STDs
(int) STDs (number)
(bool) STDs:condylomatosis
(bool) STDs:cervical condylomatosis
(bool) STDs:vaginal condylomatosis
(bool) STDs:vulvo-perineal condylomatosis
(bool) STDs:syphilis
(bool) STDs:pelvic inflammatory disease
(bool) STDs:genital herpes
(bool) STDs:molluscum contagiosum
(bool) STDs:AIDS
(bool) STDs:HIV
(bool) STDs:Hepatitis B
(bool) STDs:HPV
(int) STDs: Number of diagnosis
(int) STDs: Time since first diagnosis
(int) STDs: Time since last diagnosis
(bool) Dx:Cancer
(bool) Dx:CIN
(bool) Dx:HPV
(bool) Dx

特征属性

Cervical cancer (Risk Factors) Data Set

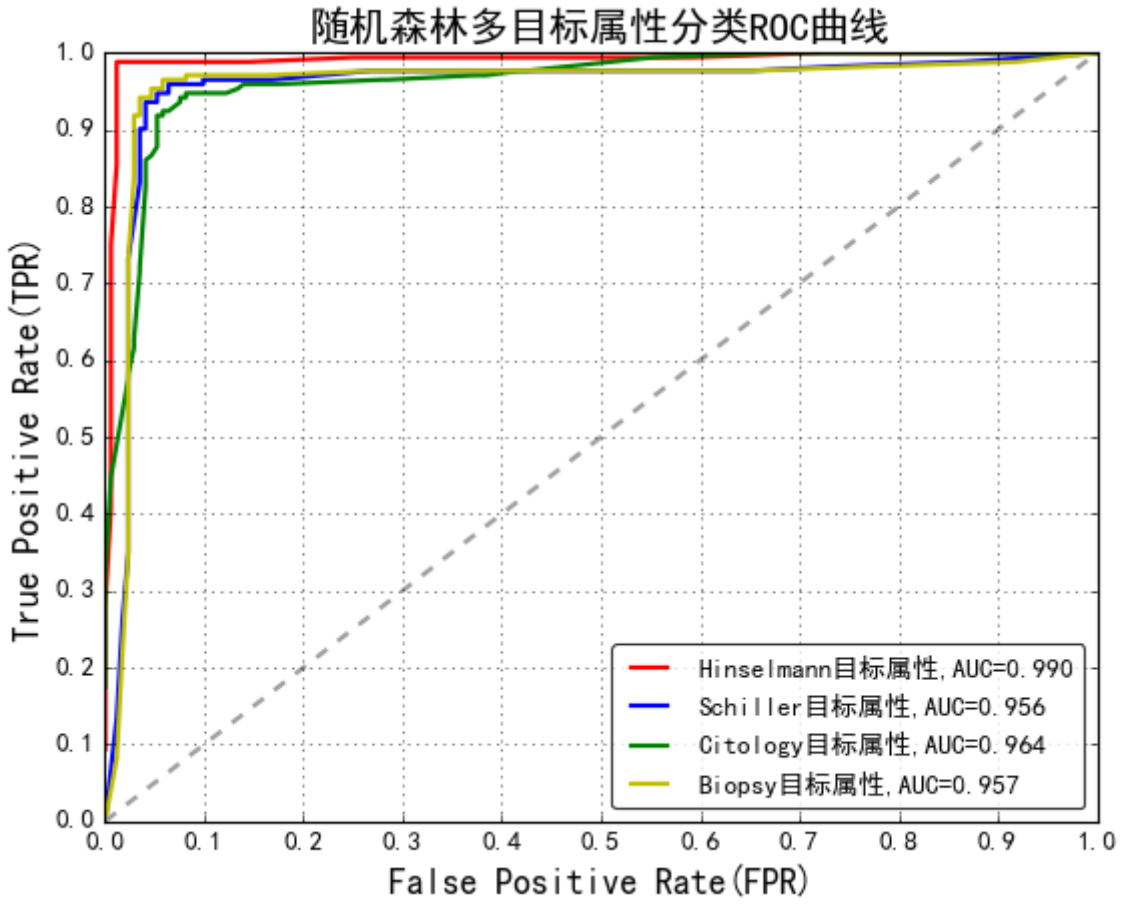
Download: Data Folder, Data Set Description

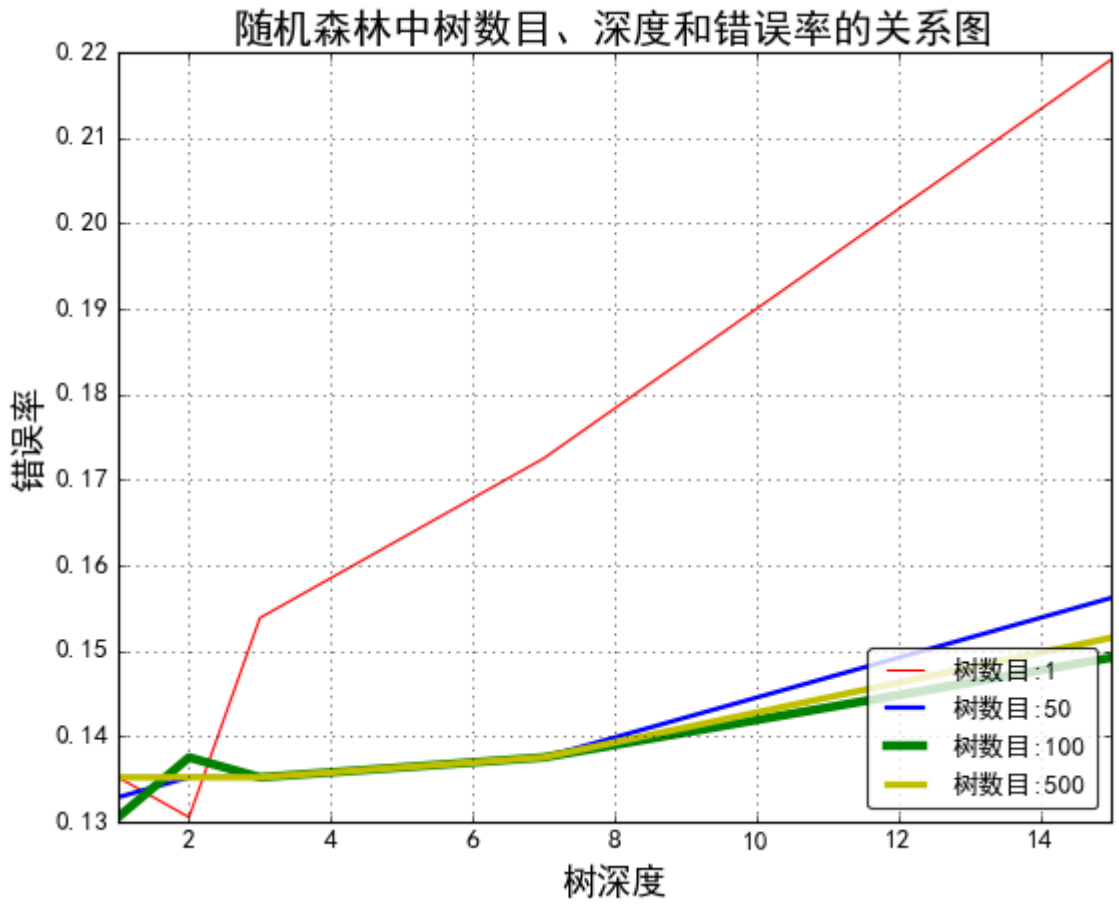
Abstract: This dataset focuses on the prediction of indicators/diagnosis of cervical cancer. The features cover demographic information, habits, and historic medical records.

Data Set Characteristics:	Multivariate	**Number of Instances:**	858	**Area:**	Life
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	36	**Date Donated**	2017-03-03
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	3687

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*) [source]

随机森林算法案例

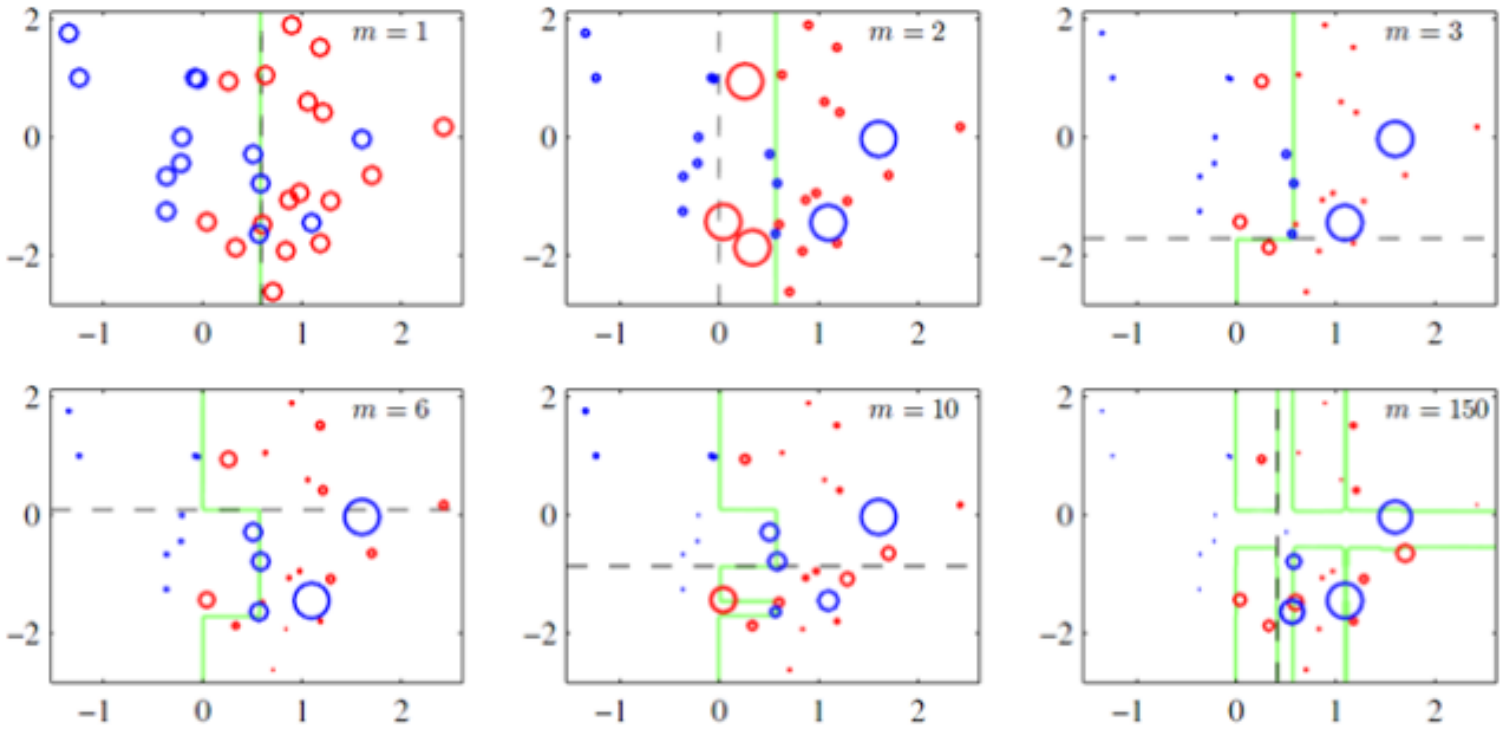

随机森林多目标属性分类ROC曲线
True Positive Rate(TPR)
False Positive Rate(FPR)
Hinselmann目标属性, AUC=0.990
Schiller目标属性, AUC=0.956
Citology目标属性, AUC=0.964
Biopsy目标属性, AUC=0.957


随机森林中树数目、深度和错误率的关系图
错误率
树深度
树数目:1
树数目:50
树数目:100
树数目:500

随机森林的思考

- 在随机森林的构建过程中，由于各棵树之间是没有关系的，相对独立的；在构建的过程中，构建第m棵子树的时候，不会考虑前面的m-1棵树。
- 思考：
  - 如果在构建第m棵子树的时候，考虑到前m-1棵子树的结果，会不会对最终结果产生有益的影响？
  - 各个决策树组成随机森林后，在形成最终结果的时候能不能给定一种既定的决策顺序呢？(也就是那颗子树先进行决策、那颗子树后进行决策)

样本加权



提升算法相关概念

- 提升是一种机器学习技术，可以用于**回归**和**分类**的问题，它每一步产生**弱预测模型**(如决策树)，并**加权累加**到总模型中；如果每一步的弱预测模型的生成都是依据损失函数的梯度方式的，那么就称为梯度提升(Gradient boosting)
- 提升技术的意义：如果一个问题存在**弱预测模型**，那么可以通过提升技术的办法得到一个**强预测模型**

梯度提升算法

- 给定输入向量X和输出变量Y组成的若干训练样本$(X_1,Y_1),(X_2,Y_2).....(X_n,Y_n)$，目标是找到近似函数F(X)，使得损失函数L(Y,F(X))的损失值最小。
- L损失函数一般采用最小二乘损失函数或者绝对值损失函数

$$L(y, F(X)) = \frac{1}{2}(y - F(X))^2 \qquad L(y, F(X)) = |y - F(X)|$$

- 最优解为：$$F^*(X) = \arg\min_F L(y, F(X))$$
- 假定F(X)是一族最优基函数$f_i(X)$的加权和：

$$F(X) = \sum_{i=1}^{M} c_i f_i(X) + const$$

F(X)函数求解推导

- 以贪心算法的思想扩展得到Fm(X)，求解最优f

$$\mathrm{F_m}(x) = F_{m-1}(x) + \arg\min_{\mathrm{f}} \sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(X_i\right) + f_m\left(X_i\right)\right)$$

- 以贪心法在每次选择最优基函数f时仍然困难，使用梯度下降的方法近似计算

- 给定常数函数$F_0(X)$

$$F_0\left(X\right) = \arg\min_{\mathrm{c}} \sum_{i=1}^{n} L\left(\mathrm{y}_i, c\right)$$

提升算法计算过程

- 计算残差

$$c_{im} = \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]$$

- 使用数据 (x_i, c_{im}) 计算拟合残差的基函数 $f_m(x)$
- 计算步长

$$c_m = \arg\min_c \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) - c \cdot f_m(x_i)))$$

- 更新模型

$$F_m(X) = F_{m-1}(X) - c_m f_m(x)$$

梯度提升迭代决策树GBDT(MART)

- GBDT由三部分构成：DT(Regression Decistion Tree)、GB(Gradient Boosting)和Shrinkage
- 由多棵决策树组成，所有树的结果累加起来就是最终结果
- 迭代决策树和随机森林的区别：
  - 随机森林使用抽取不同的样本构建不同的子树，也就是说第m棵树的构建和前m-1棵树的结果是没有关系的
  - 迭代决策树在构建子树的时候，使用之前子树构建结果后形成的残差作为输入数据构建下一个子树；然后最终预测的时候按照子树构建的顺序进行预测，并将预测结果相加

Adaboost算法

- 给定样本数据集的样本权重，然后进行预测模型的构建，同时根据是否预测是否增强，修改权重值，迭代创建模型，直到错误率为0或者达到停止条件。
- Adaboost算法将基分类器的线性组合作为强分类器，同时给分类误差率较小的基本分类器以大的权值，给分类误差率较大的基分类器以小的权重值；构建的线性组合为：

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

- 最终分类器为：

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

Adaboost算法

- 假设训练数据集T={$(X_1,Y_1),(X_2,Y_2)....(X_n,Y_n)$}
- 初始化训练数据权重分布

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

- 使用具有权值分布Dm的训练数据集学习，得到基本分类器

$$G_m(x) : X \rightarrow \{-1,1\}$$

- 计算Gm(x)在训练集上的分类误差 $\varepsilon_m = \sum_{i=1}^{N} W_{mi} I(G_m(x_i) \neq y_i)$
- Adaboost算法的alpha参数值为： $\alpha_m = \frac{1}{2} * \log\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$

Adaboost算法

- 权重训练数据集的权值分布

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

$$W_{m+1,i} = \frac{W_{m,i}}{Z_m} e^{-\alpha_m Y_i G_m(X_i)}$$

- 这里Zm是规范化因子 $Z_m = \sum_{i=1}^{N} W_{mi} e^{-\alpha_m Y_i G_m(X_i)}$

- 构建基本分类器的线性组合 $f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$

- 得到最终分类器 $G(x) = sign(f(x)) = sign(\sum_{m=1}^{M} \alpha_m G_m(x))$

Adaboost算法的直观理解

- 权重训练用下列训练样本，试用AdaBoost算法学习一个强分类器

序号	1	2	3	4	5	6	7	8	9	X
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	−1	−1	−1	1	1	1	−1

- 初始化训练数据集的权值分布

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

$$w_{1i} = 0.1$$

Adaboost算法的直观理解

- 对于m=1

序号	1	2	3	4	5	6	7	8	9	X
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1

- 在权值分布为D1的训练数据上，阈值v取2.5时误差率最低，故基本分类器为：

$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$

- G1(x)在训练数据集上的误差率 $\varepsilon_1 = P(G1(xi) \neq yi) = 0.3$

- 计算G1的系数 $\alpha_1 = \frac{1}{2}\log\frac{1-\varepsilon_1}{\varepsilon_1} = 0.4236$

Adaboost算法的直观理解

- 更新数据集的权值分布 $D_{m+1} = (w_{m+1,1}, w_{m+1,2}, \cdots w_{m+1,i} \cdots, w_{m+1,n})$

$$W_{m+1,i} = \frac{W_{m,i}}{Z_m} e^{-\alpha_m Y_i G_m(X_i)}$$

$$D2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715)$$

$$f_1(x) = 0.4236 G_1(x)$$

- 分类器sign(f1(x))在训练数据集上有3个误分类点

Adaboost算法的直观理解

- 对于m=2

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w	0.0715	0.0715	0.0715	0.0715	0.0715	0.0715	0.1666	0.1666	0.1666	0.0715

- 在权值分布为D2的训练数据上，阈值v取8.5时误差率最低，故基本分类器为：

$$G_2(x)=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

- G2(x)在训练数据集上的误差率

$$\varepsilon_2 = P(G_2(xi) \neq yi) = 0.2143(0.0715 * 3)$$

Adaboost算法的直观理解

- 计算G2的系数 $\alpha_2 = \frac{1}{2}\log\frac{1-\varepsilon_2}{\varepsilon_2} = 0.6496$
- 更新数据集的权值分布 $D_{m+1} = (w_{m+1,1}, w_{m+1,2}, \ldots w_{m+1,i} \ldots, w_{m+1,n})$

$$W_{m+1,i} = \frac{W_{m,i}}{Z_m} e^{-\alpha_m Y_i G_m(X_i)}$$

$$D3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.1667, 0.1060, 0.1060, 0.1060, 0.0455)$$

$$f_2(x) = 0.4236 G_1(x) + 0.6496 G_2(x)$$

- 分类器sign(f2(x))在训练数据集上有3个误分类点

Adaboost算法的直观理解

- 对于m=3

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1060	0.1060	0.1060	0.0455

- 在权值分布为D3的训练数据上，阈值v取5.5时误差率最低，故基本分类器为：

$$G_3(x)=\begin{cases}1, & x>5.5\\ -1, & x<5.5\end{cases}$$

- G3(x)在训练数据集上的误差率

$$\varepsilon_3 = P(G_3(xi) \neq yi) = 0.1820(0.0455 * 4)$$

Adaboost算法的直观理解

- 计算G3的系数 $\alpha_3 = \frac{1}{2}\log\frac{1-\varepsilon_3}{\varepsilon_3} = 0.7514$
- 更新数据集的权值分布 $D_{m+1} = (w_{m+1,1}, w_{m+1,2}, \ldots w_{m+1,i} \ldots, w_{m+1,n})$

$$W_{m+1,i} = \frac{W_{m,i}}{Z_m} e^{-\alpha_m Y_i G_m(X_i)}$$

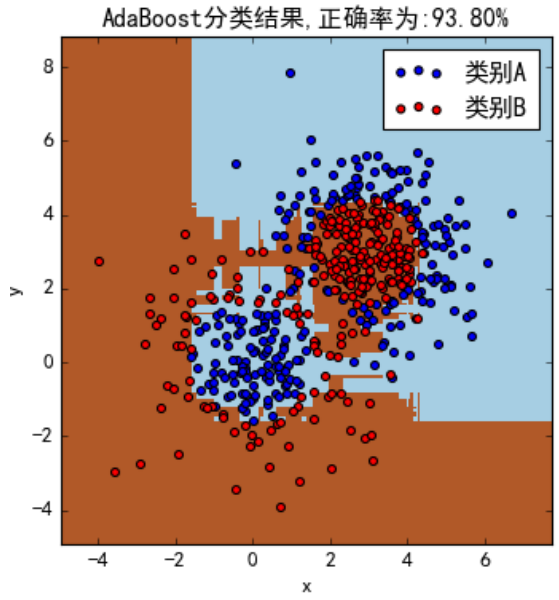
$$D4 = (0.125, 0.25, 0.125, 0.102, 0.102, 0.102, 0.065, 0.065, 0.065, 0.125)$$

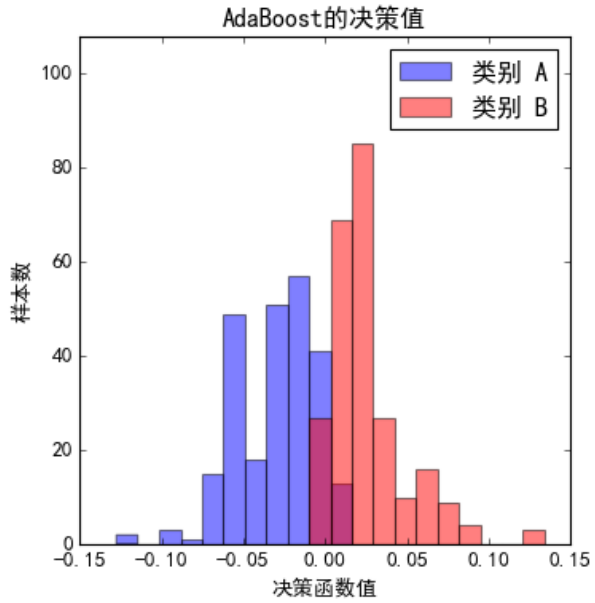
$$f_3(x) = 0.4236 G_1(x) + 0.6496 G_2(x) + 0.7514 G_3(x)$$

- 分类器sign(f3(x))在训练数据集上有0个误分类点

Adaboost案例一

- 基于python的sklearn模块中的API创建模拟数据，并进行Adaboost API进行数据分类开发测试



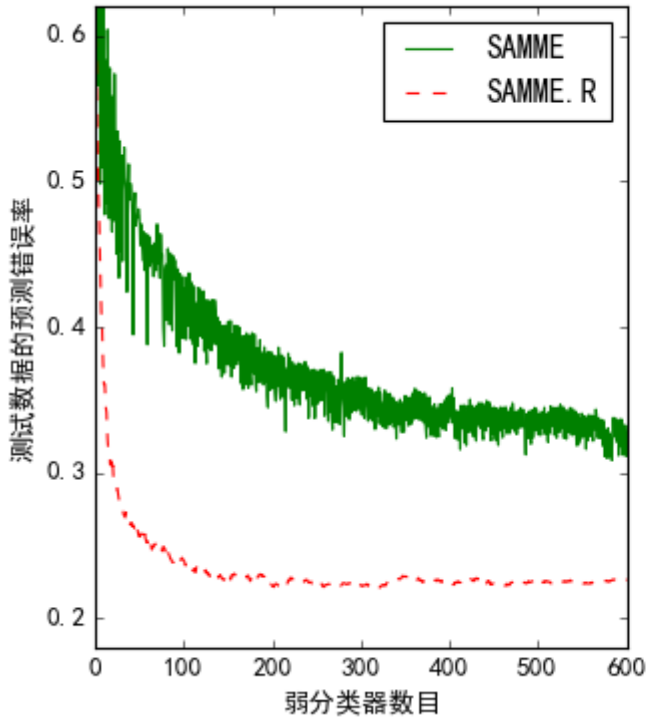


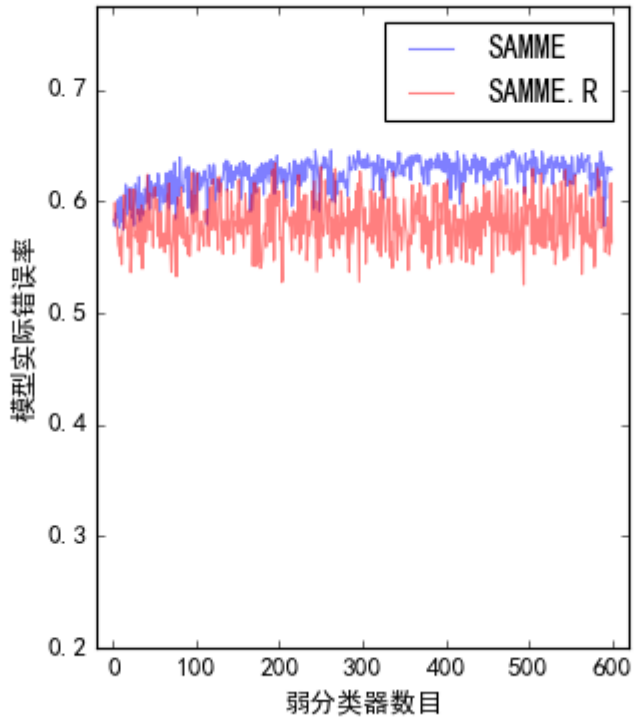
sklearn.datasets.**make_gaussian_quantiles**(*mean=None, cov=1.0, n_samples=100, n_features=2, n_classes=3, shuffle=True, random_state=None*) ¶ [source]

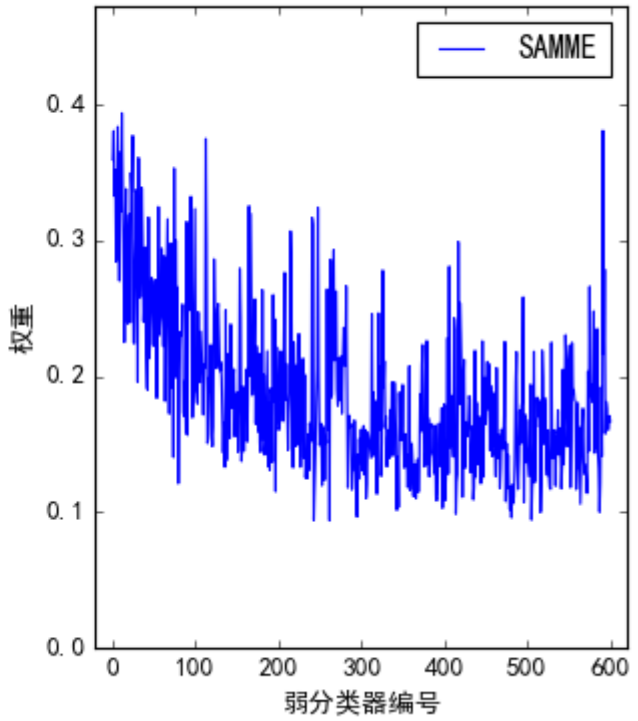
class sklearn.ensemble.**AdaBoostClassifier**(*base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None*) [source]

Adaboost案例二

- 基于python的sklearn模块中的API创建模拟数据，Adaboost API的参数algorithm的不同取值进行测试，并得到比较效果值

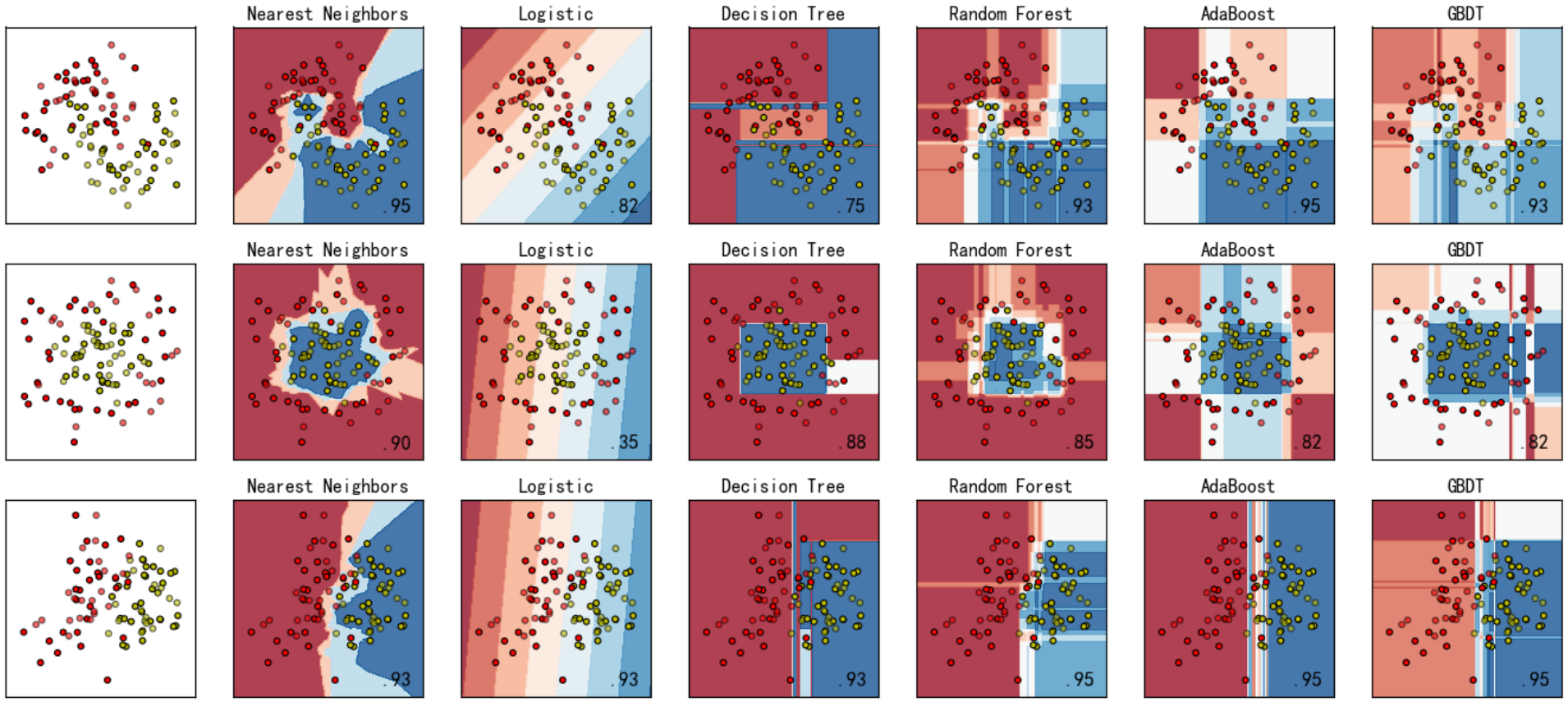






综合案例：分类算法比较

- 比较逻辑回归、KNN、决策树、随机森林、GBDT、Adaboost等分类算法的效果，数据集使用sklearn自带的模拟数据进行测试。



THANK YOU

上海育创网络科技有限公司