北风网
IBEIFENG.COM

人工智能之机器学习

决策树

上海育创网络科技股份有限公司

主讲人：刘老师(GerryLiu)

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”，不推荐就业

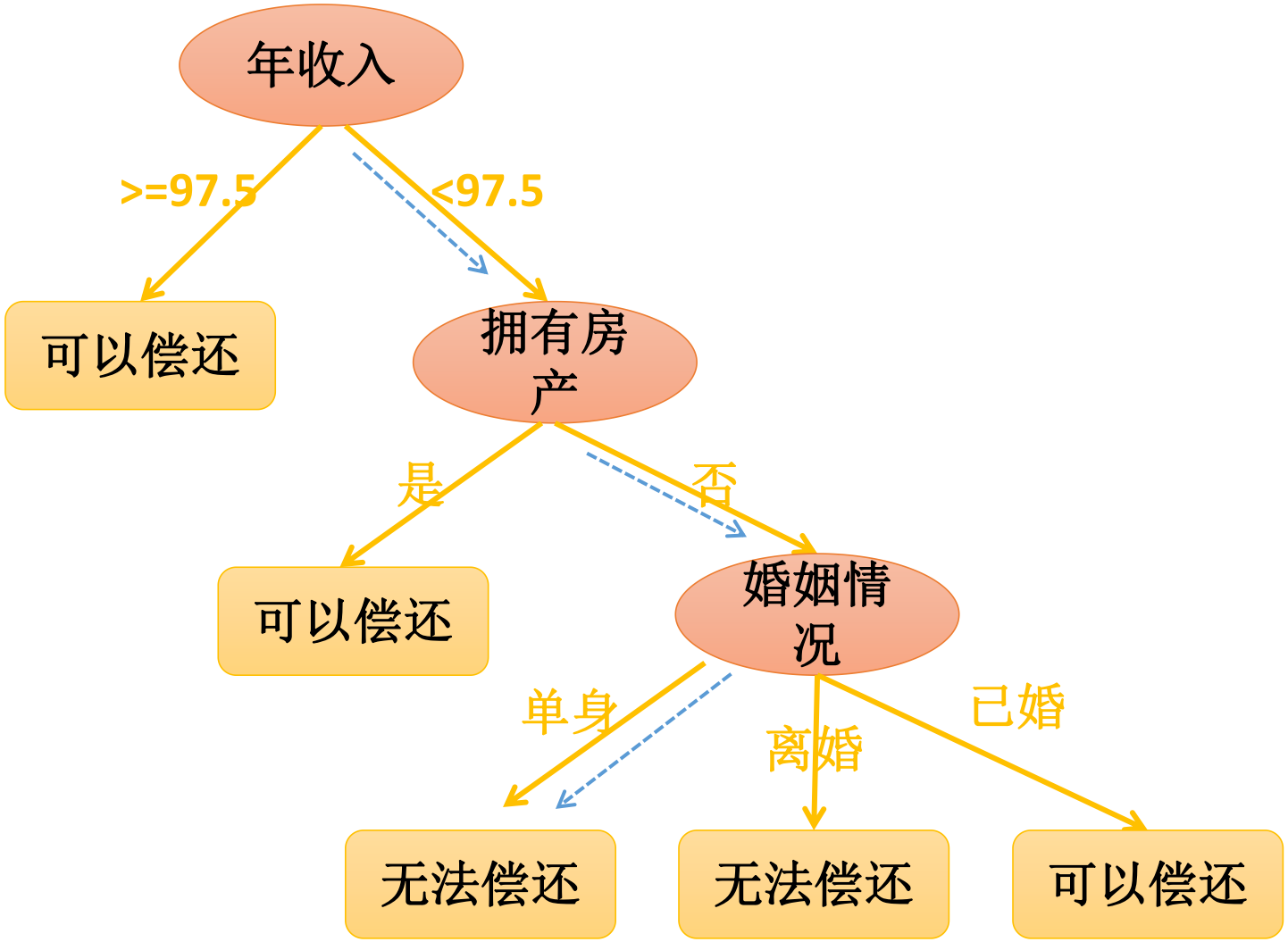
课程内容

- 信息熵
- 决策树
- 决策树优化
- 剪枝
- 决策树可视化

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



- 当构建好一个判断模型后，新来一个用户后，可以根据构建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。

比特化(Bits)

- 假设存在一组随机变量X，各个值出现的概率关系如图；
- 现在有一组由X变量组成的序列: BACADDCBAC.....；如果现在希望将这个序列转换为二进制来进行网络传输，那么我们得到一个得到一个这样的序列: 01001000111110010010.....
- 结论: 在这种情况下，我们可以使用两个比特位来表示一个随机变量。

P(X=A)=1/4	P(X=B)=1/4	P(X=C)=1/4	P(X=D)=1/4

A	B	C	D
00	01	10	11

比特化(Bits)

- 而当X变量出现的概率值不一样的时候，对于一组序列信息来讲，每个变量平均需要多少个比特位来描述呢??

P(X=A)=1/2	P(X=B)=1/4	P(X=C)=1/8	P(X=D)=1/8

A	B	C	D
0	10	110	111

$$E = 1*\frac{1}{2} + 2*\frac{1}{4} + 3*\frac{1}{8} + 3*\frac{1}{8} = 1.75$$

$$E = -\log_2\left(\frac{1}{2}\right)*\frac{1}{2} - \log_2\left(\frac{1}{4}\right)*\frac{1}{4} - \log_2\left(\frac{1}{8}\right)*\frac{1}{8} - \log_2\left(\frac{1}{8}\right)*\frac{1}{8} = 1.75$$

一般化的比特化(Bits)

- 假设现在随机变量X具有m个值，分别为: $V_1,V_2,....,V_m$；并且各个值出现的概率如下表所示；那么对于一组序列信息来讲，每个变量平均需要多少个比特位来描述呢??

$P(X=V_1)=p_1$	$P(X=V_2)=p_2$	$P(X=V_3)=p_3$	.....	$P(X=V_m)=p_m$

- 可以使用这些变量的期望来表示每个变量需要多少个比特位来描述信息:

$$E(X) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) - ... - p_m \log_2(p_m)$$

$$= -\sum_{i=1}^{m} p_i \log_2(p_i)$$

信息熵(Entropy)

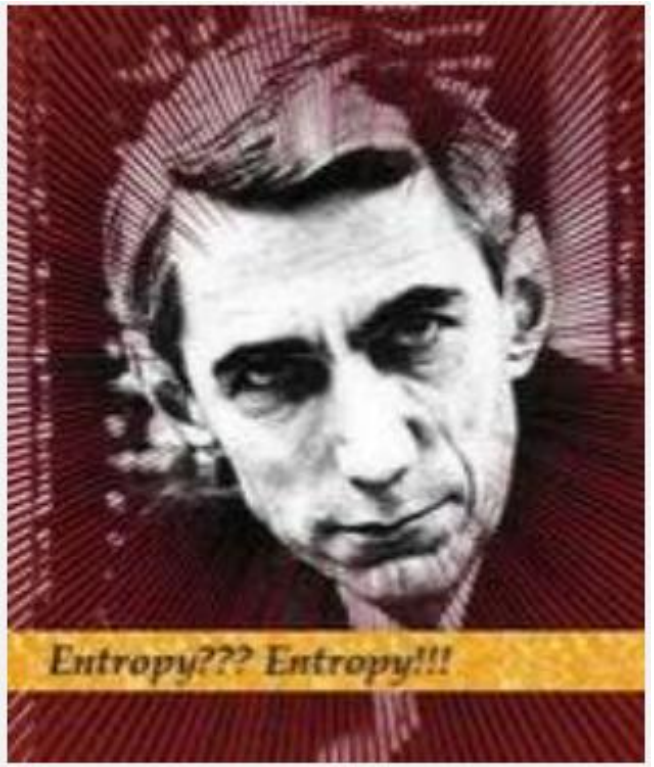
- H(X)就叫做随机变量X的信息熵；

$$H(X)=-\sum_{i=1}^{m} p_i \log_2(p_i)$$

信息熵(Entropy)

$$H(X) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

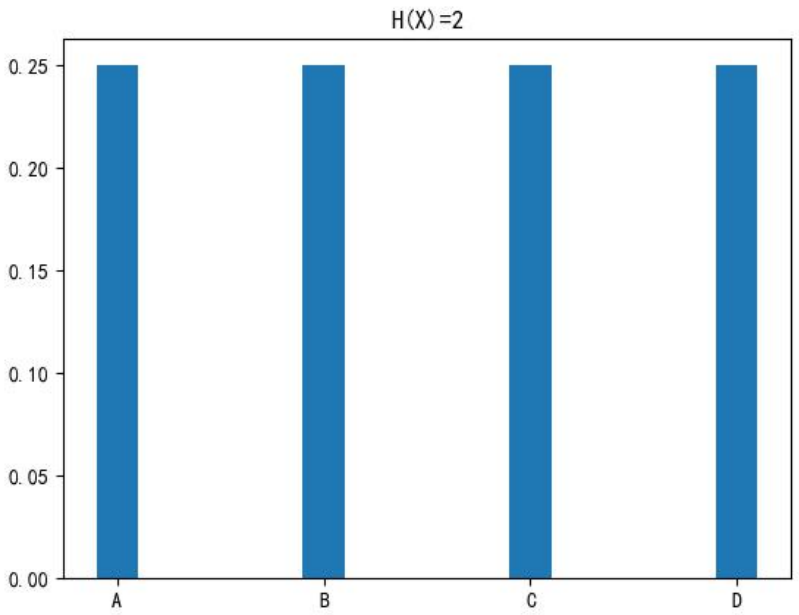
- 信息量：指的是一个样本/事件所蕴含的信息，如果一个事件的概率越大，那么就可以认为该事件所蕴含的信息越少。极端情况下，比如：“太阳从东方升起”，因为是确定事件，所以不携带任何信息量。
- 信息熵：1948年，香农引入信息熵；一个系统越是有序，信息熵就越低，一个系统越是混乱，信息熵就越高，所以信息熵被认为是一个系统有序程度的度量。
- **信息熵就是用来描述系统信息量的不确定度。**

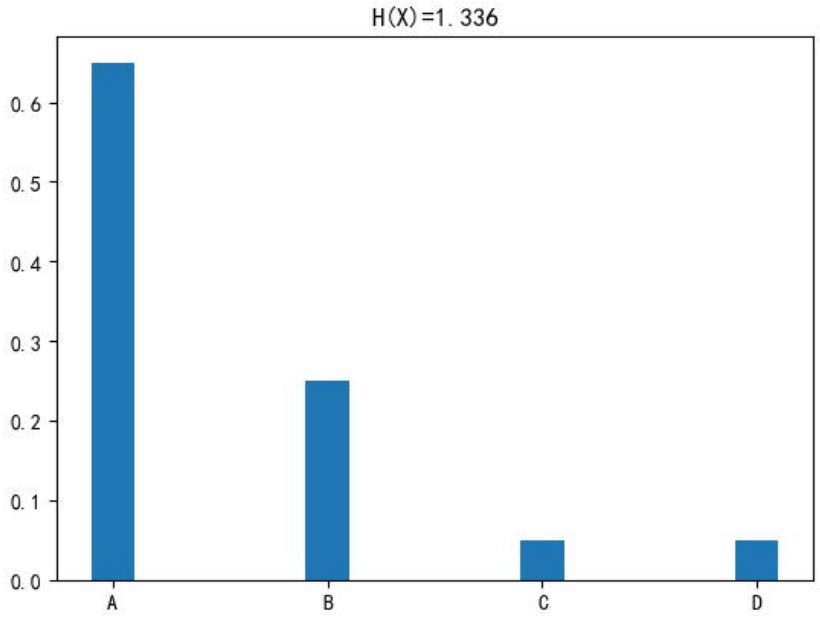


信息熵(Entropy)

$$H(X)=-\sum_{i=1}^{m} p_i \log_2(p_i)$$

- High Entropy(高信息熵)：表示随机变量X是均匀分布的，各种取值情况是等概率出现的。
- Low Entropy(低信息熵)：表示随机变量X各种取值不是等概率出现。可能出现有的事件概率很大，有的事件概率很小。





信息熵(Entropy)案例

$$H(X)=-\sum_{i=1}^{m} p_i \log_2(p_i)$$

- 赌马比赛中，有两组赛马共八匹，获胜的概率如下：

	第一组	第二组
P(X=A)	1/4	13/20
P(X=B)	1/4	5/20
P(X=C)	1/4	1/20
P(X=D)	1/4	1/20

- 在比赛前，对于第一组而言，我们只知道A/B/C/D获胜的概率是一样的，我们是判断不出来任何偏向的；但是对于第二组而言，我们很清楚的就能够判断A会获胜。

条件熵H(Y|X)

- 给定条件X的情况下，随机变量Y的信息熵就叫做条件熵。

专业(X)	性别(Y)
数学	M
IT	M
英语	F
数学	F
数学	M
IT	M
英语	F
数学	F

$$P(X=\text{数学})=0.5$$
$$P(Y=M)=0.5$$
$$P(X=\text{数学},Y=F)=0.25$$
$$P(Y=M\mid X=\text{英语})=0$$

$$H(X)=1.5$$
$$H(Y)=1$$

条件熵H(Y|X)

- 当专业(X)为数学的时候，Y的信息熵的值为:H(Y|X=数学)

专业(X)	性别(Y)
数学	M
IT	M
英语	F
数学	F
数学	M
IT	M
英语	F
数学	F

专业(X)	性别(Y)
数学	M
数学	F
数学	M
数学	F

$$H(Y \mid X = 数学) = 1$$

条件熵H(Y|X)

- 给定条件X的情况下，所有不同x值情况下Y的信息熵的平均值叫做条件熵。

专业(X)	性别(Y)
数学	M
IT	M
英语	F
数学	F
数学	M
IT	M
英语	F
数学	F

$$H(Y \mid X)=\sum_{j=1} P(X=v_j)H(Y \mid X=v_j)$$

v_j	$P(X=v_j)$	$H(Y \mid X=v_j)$
数学	0.5	1
IT	0.25	0
英语	0.25	0

$$H(Y \mid X)=0.5*1+0.25*0+0.25*0=0.5$$

条件熵H(Y|X)

- 给定条件X的情况下，所有不同x值情况下Y的信息熵的平均值叫做条件熵。另外一个公式如下所示：

$$H(Y \mid X) = H(X,Y) - H(X)$$

- 事件(X,Y)发生所包含的熵，减去事件X单独发生的熵，即为在事件X发生的前提下，Y发生“新”带来的熵，这个也就是条件熵本身的概念。

条件熵H(Y|X)

$$H(Y|X)=\sum_{j=1}P(\mathrm{X}=\mathrm{v_j})H(Y|X=v_j)=\sum_{x}P(\mathrm{x})H(Y|x)$$

$$=\sum_{x}p(x)\left(-\sum_{y}p(y|x)\log(p(y|x))\right)=-\sum_{x}\sum_{y}p(x)p(y|x)\log(p(y|x))$$

$$=-\sum_{x}\sum_{y}p(x,y)\log\left(\frac{p(x,y)}{p(x)}\right)$$

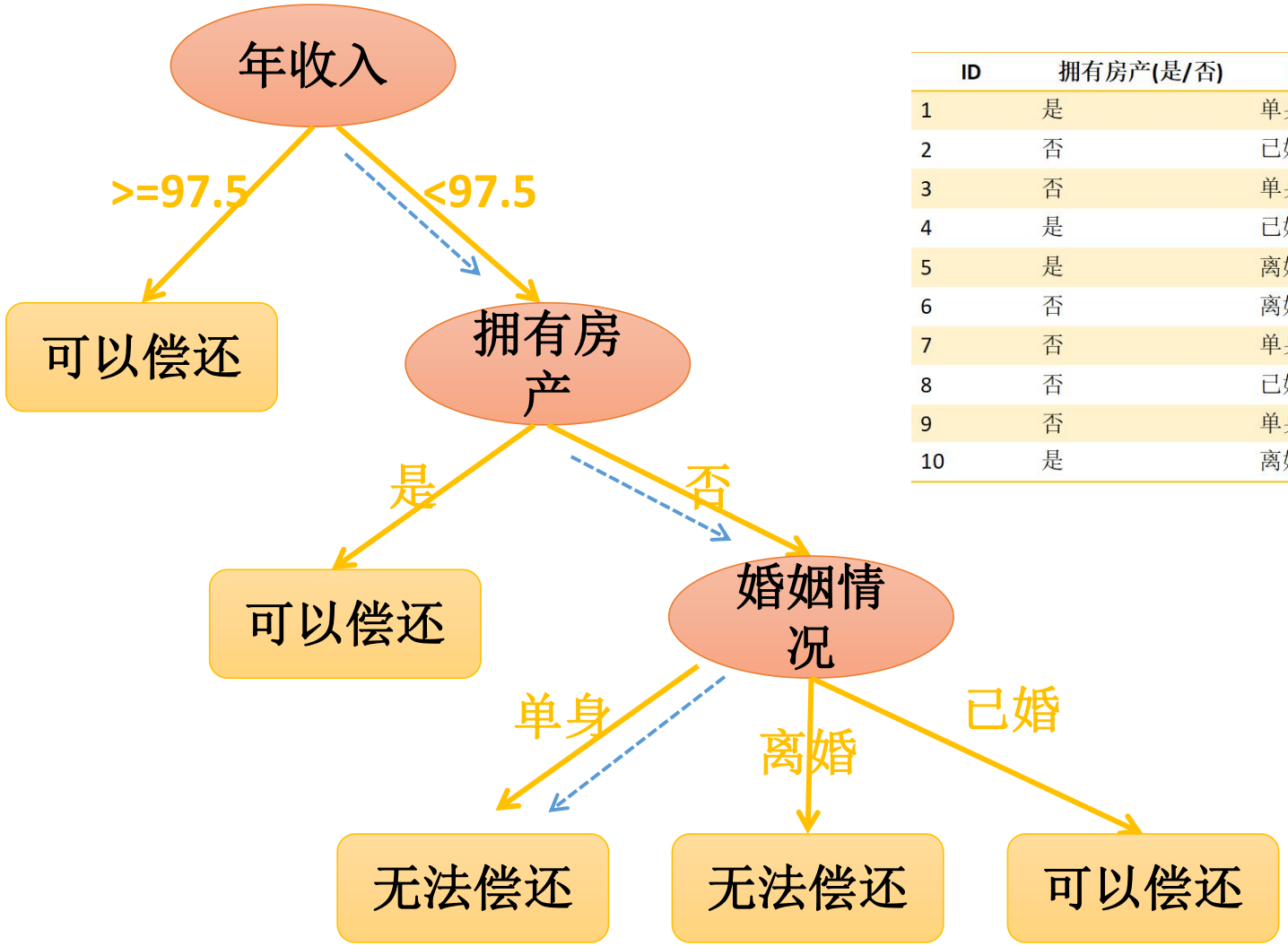
$$=-\sum_{x}\sum_{y}p(x,y)\log(p(x,y))-\left[-\sum_{x}\left(\sum_{y}p(x,y)\right)\log(p(x))\right]$$

$$=H(X,Y)-\left[-\sum_{x}p(x)\log(p(x))\right]=H(X,Y)-H(X)$$

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，每个分支表示一个测试输出，每个叶节点代表一种预测类别；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

决策树构建



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'(让一个分裂子数据集中待分类的项尽可能的属于同一个类别)。
- 构建步骤如下:
  - 1. 将所有的特征看成一个一个的节点；
  - 2. 遍历当前特征的每一种分割方式，找到最好的分割点；将数据划分为不同的子节点，eg： N_1、N_2....N_m；计算划分之后所有子节点的'纯度'信息；
  - 3. 使用第二步遍历所有特征，选择出最优的特征以及该特征的最优的划分方式；得出最终的子节点: N_1、N_2....N_m
  - 4. 对子节点N_1、N_2....N_m分别继续执行2-3步，直到每个最终的子节点都足够'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，直到得到最终结果。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：

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

- D目标属性，A为某一个待划分的特征属性；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

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
  - 当前节点中样本数小于某个阈值，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
- NOTE：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不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−) = $\frac{\text{FNR}}{\text{TNR}}$	

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

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)

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

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

$$H(Y \mid X)=\sum_{j=1} P(\mathrm{X}=\mathrm{v}_{\mathrm{j}})H(Y \mid X=v_j)$$

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

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

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

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

第一个分割属性为收入

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

X拥有房产(是/否)	Y无法偿还债务(是/否)
是	否
否	否
否	否
是	否
是	否
否	是
否	是
否	否
否	是
是	否

$$p(x=是)=0.4 \qquad p(x=否)=0.6$$

$$p(y=是\mid x=是)=0 \qquad p(y=否\mid x=是)=1$$

$$p(y=是\mid x=否)=0.5 \qquad p(y=否\mid x=否)=0.5$$

$$Info(D_{有房产})=H(D_{有房产})=-1*\log_2(1)-0*\log_2(0)=0$$

$$Info(D_{无房产})=H(D_{无房产})=-0.5\log_2(0.5)-0.5\log_2(0.5)=1$$

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

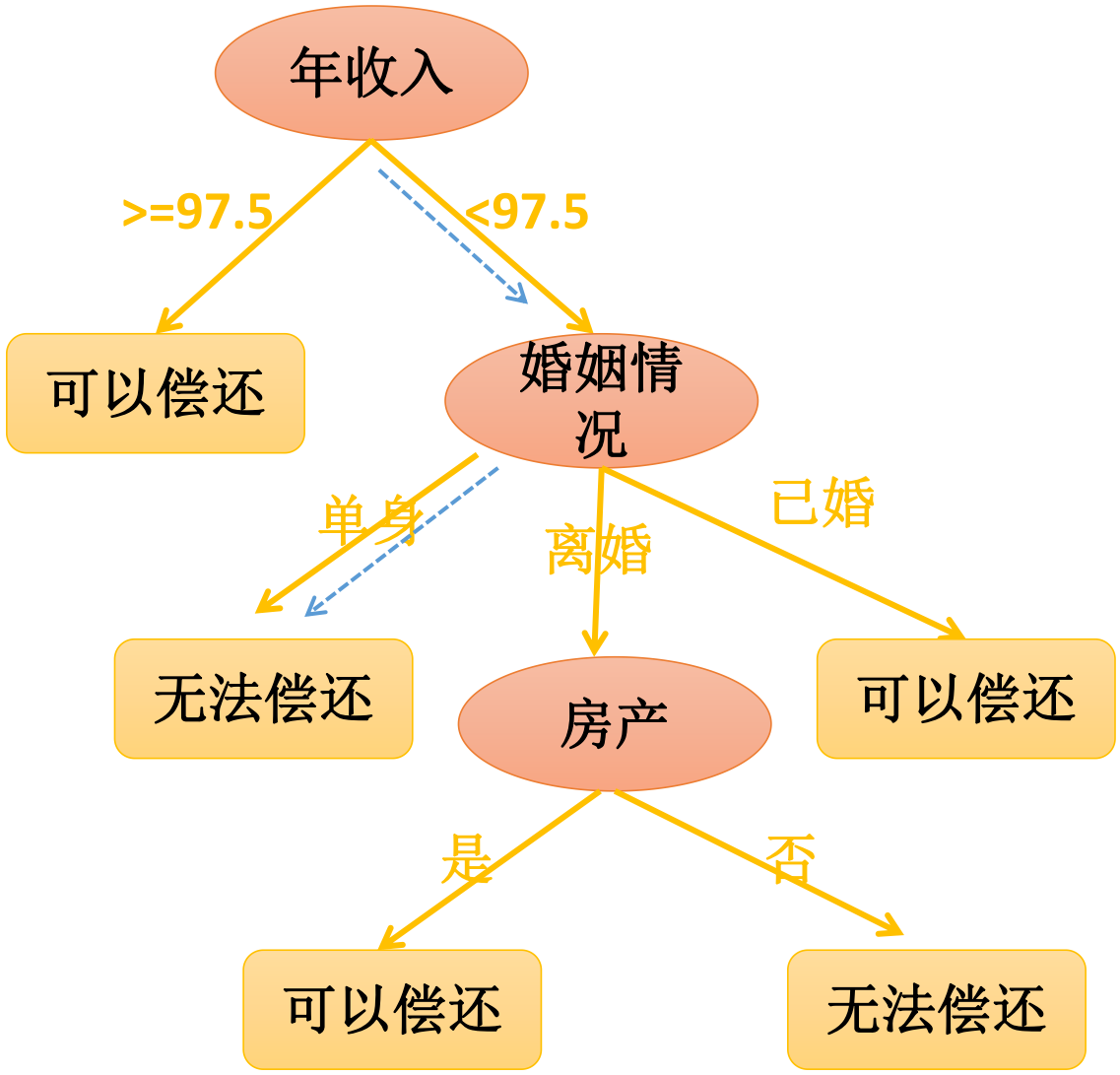
决策树直观理解结果计算

X(年收入)	60	75	85	90	95	100	100	110	125	220
Y(无法偿还债务)	否	否	是	是	是	否	否	否	否	否
分割点	~~67.5~~	80	~~87.5~~	~~92.5~~	97.5	~~100~~	~~105~~	~~117.5~~	~~172.5~~	

$$Gain(\text{收入} = 97.5) = 0.395$$

$$Gain(\text{收入} = 80) = 0.116$$

决策树直观理解



- 重新构建决策树!!!(如果一个特征不允许重复选择)
- 当构建好一个判断模型后，新来一个用户后，可以根据构建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。

决策树主要算法

- 建立决策树的主要是以下三种算法
  - ID3
  - C4.5
  - CART（Classification And Regression Tree）

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
  - 1. ID3算法只支持离散的特征属性，不支持连续的特征属性
  - 2. ID3算法构建的是多叉树

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

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

ID3算法优缺点

- 优点:
  - 决策树构建速度快；实现简单；
- 缺点:
  - 计算依赖于特征取值数目较多的特征，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**信息增益率**来取代ID3算法中的信息增益，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构建的是多分支的决策树；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

CART算法

- 使用**基尼系数(分类树)**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。强调备注：**CART构建是二叉树**。

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

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

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

ID3、C4.5、CART分类树算法总结

- ID3、C4.5和CART算法均只适合在小规模数据集上使用
- ID3、C4.5和CART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算法是三种算法中最常用的一种决策树构建算法(sklearn中仅支持CART)。
- 三种算法的区别仅仅只是对于当前树的评价标准不同而已，ID3使用**信息增益**、C4.5使用**信息增益率**、CART使用**基尼系数**。（不是主要区别）
- CART算法构建的一定是二叉树，ID3和C4.5构建的不一定是二叉树。(主要区别)

ID3、C4.5、CART分类树算法总结

算法	支持模型	树结构	划分特征选择	连续值处理	缺失值处理	剪枝	特征属性多次使用
ID3	分类	多叉树	信息增益	不支持	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益率	支持	支持	支持	不支持
CART	分类、回归	二叉树	基尼系数、均方差	支持	支持	支持	支持

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

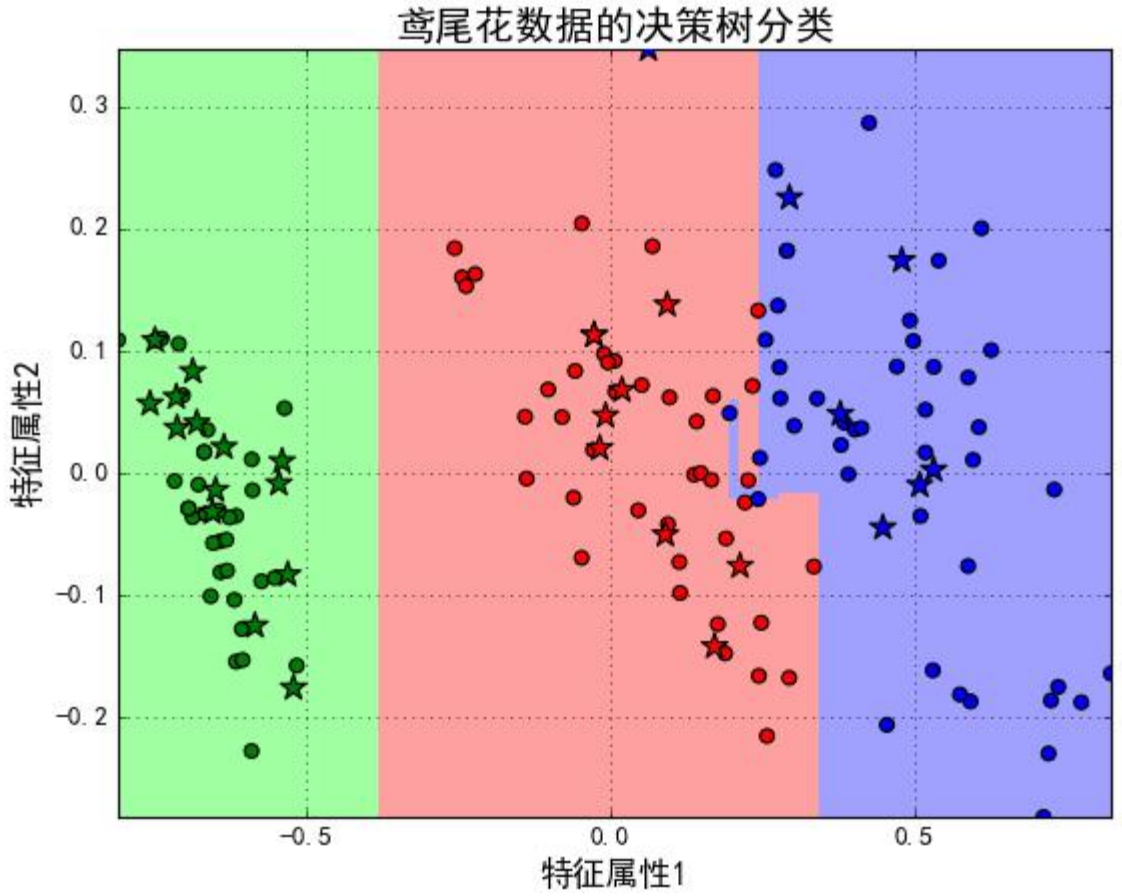
Download: Data Folder, Data Set Description

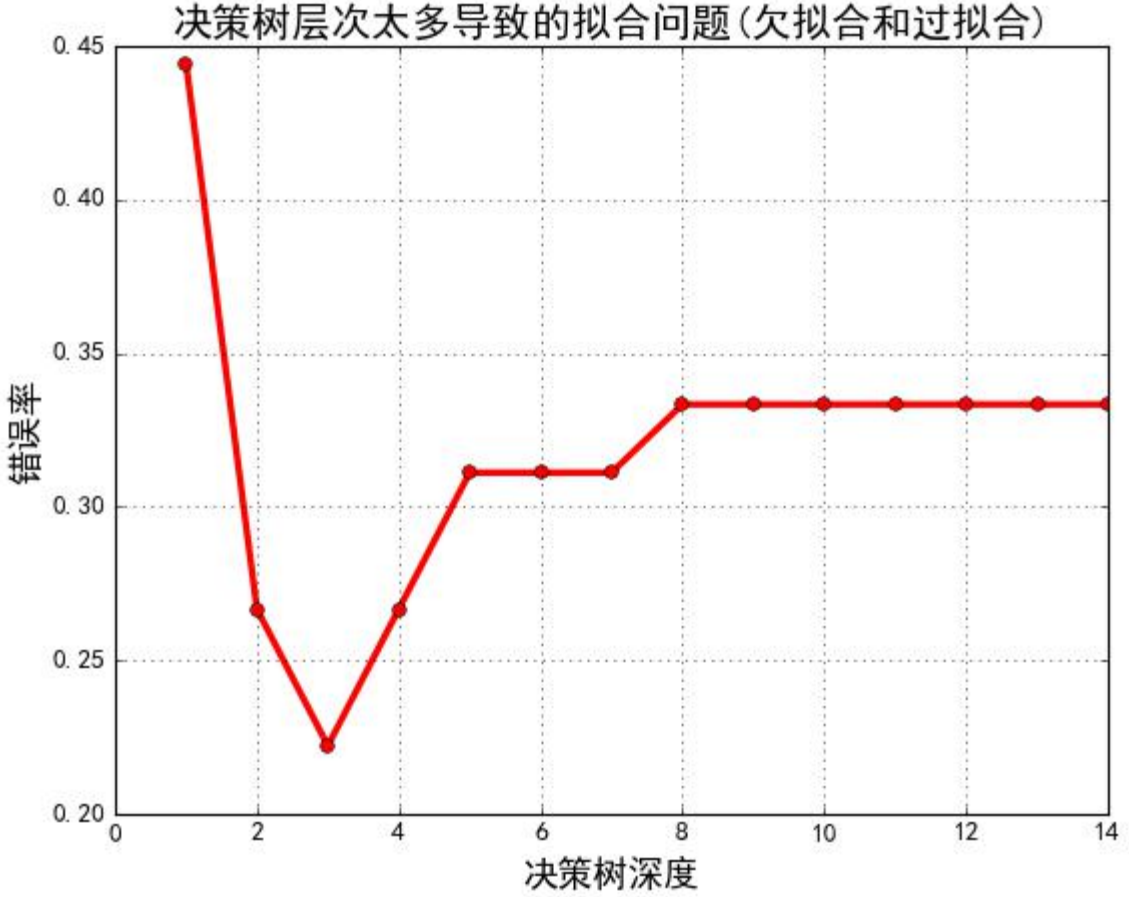
Abstract: Famous database; from Fisher, 1936

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

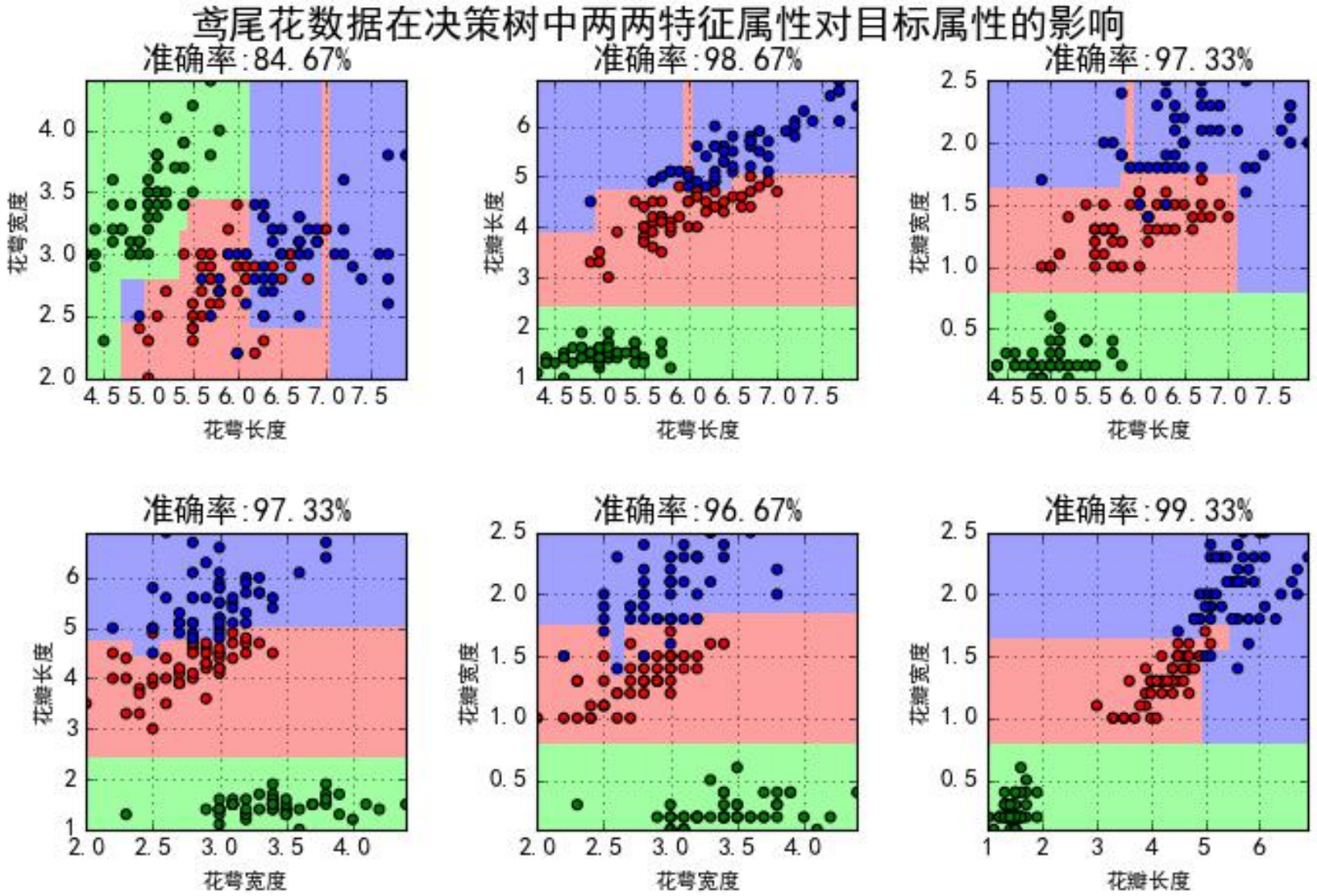
class sklearn.tree.**DecisionTreeClassifier**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, class_weight=None*) ¶ [source]

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



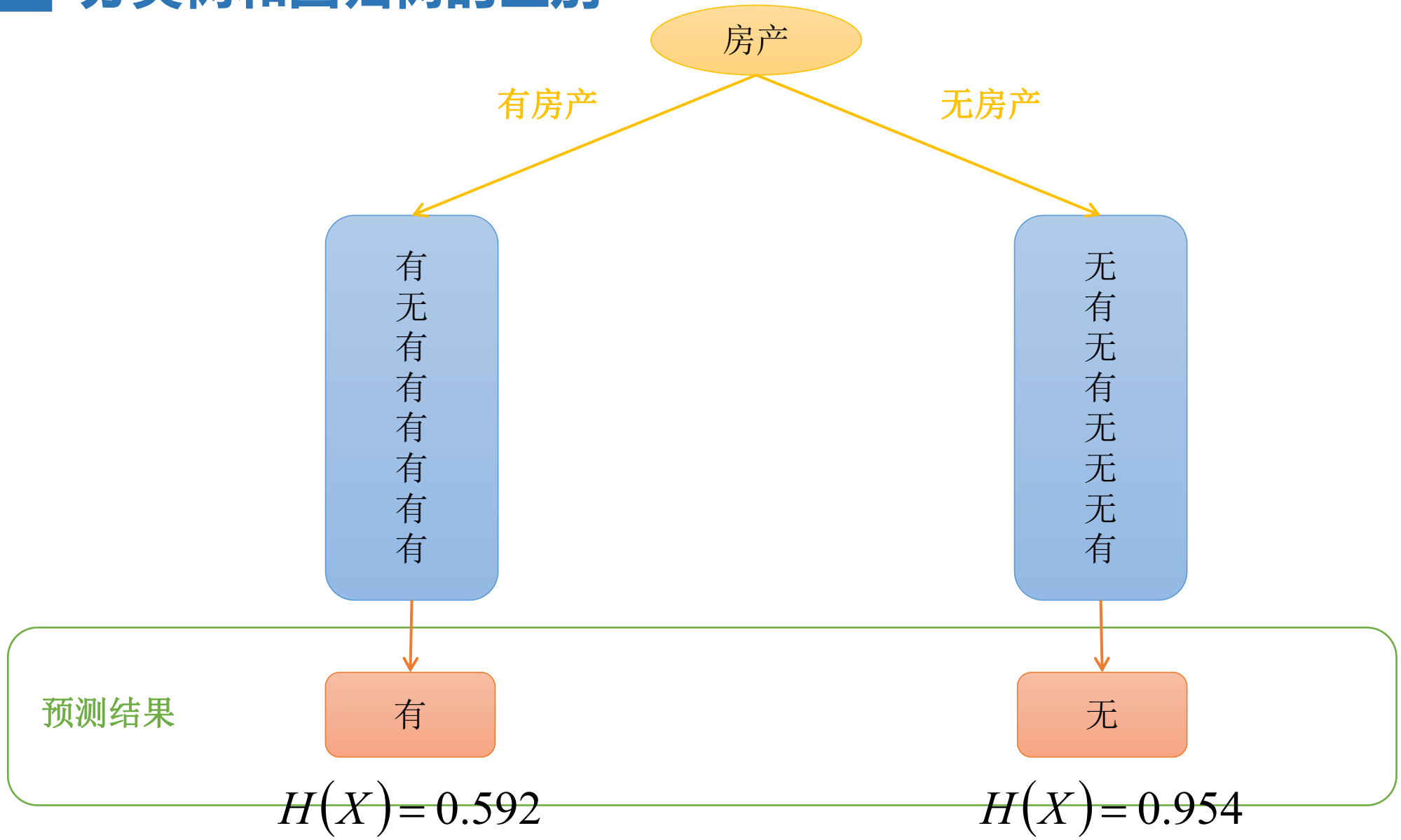
分类树和回归树的区别

- 分类树采用信息增益、信息增益率、基尼系数来评价树的效果，都是基于概率值进行判断的；而分类树的叶子节点的预测值一般为叶子节点中概率最大的类别作为当前叶子的预测值。
- 在回归树中，叶子节点的预测值一般为叶子节点中所有值的均值来作为当前叶子节点的预测值。所以在回归树中一般采用MSE或者MAE作为树的评价指标，即均方差。

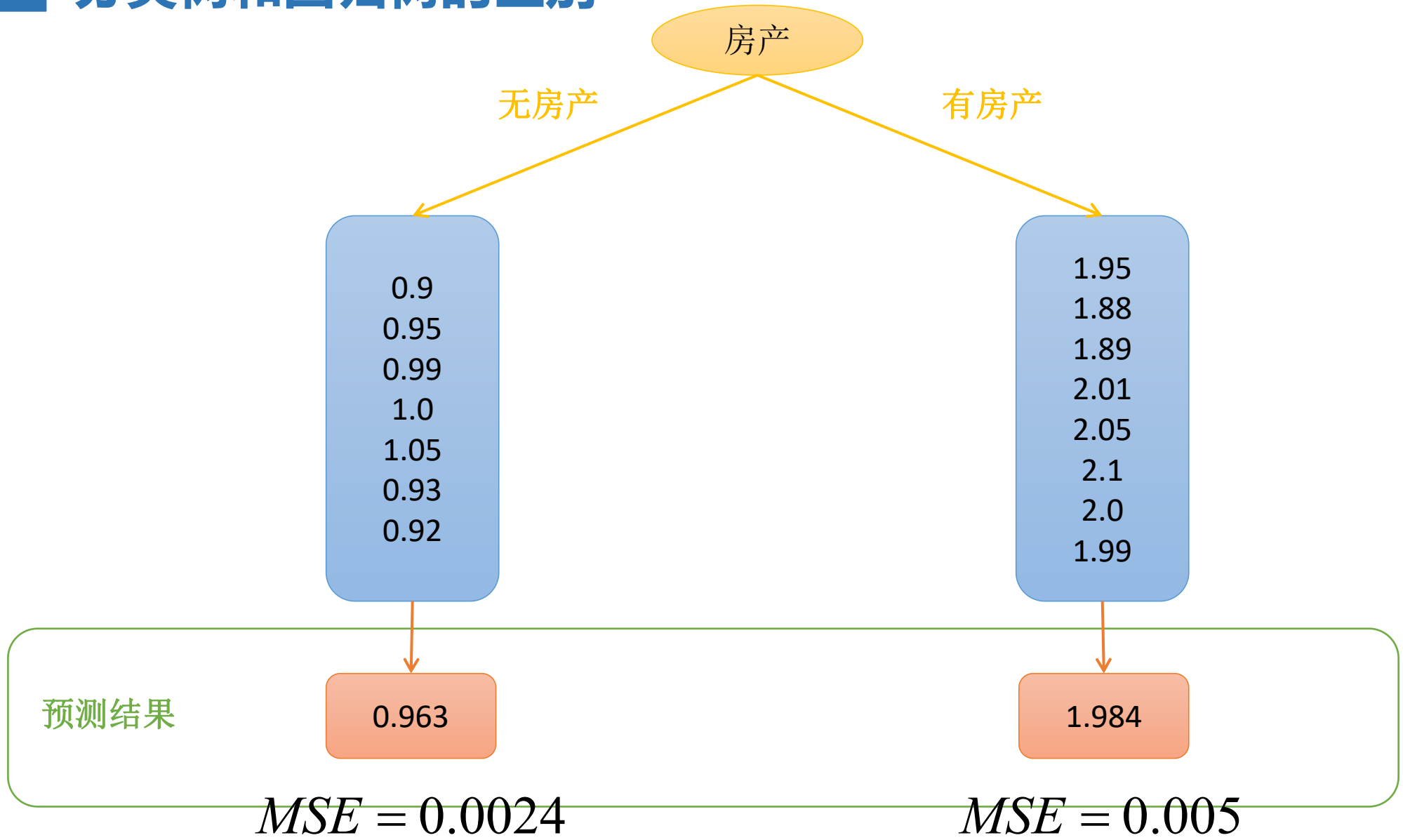
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- 一般情况下，只会使用CART算法构建回归树。

分类树和回归树的区别



分类树和回归树的区别


房产
无房产
有房产
0.9
0.95
0.99
1.0
1.05
0.93
0.92
1.95
1.88
1.89
2.01
2.05
2.1
2.0
1.99
预测结果
0.963
1.984
$MSE = 0.0024$
$MSE = 0.005$

决策树案例二：波士顿房屋租赁价格预测(作业)

- 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

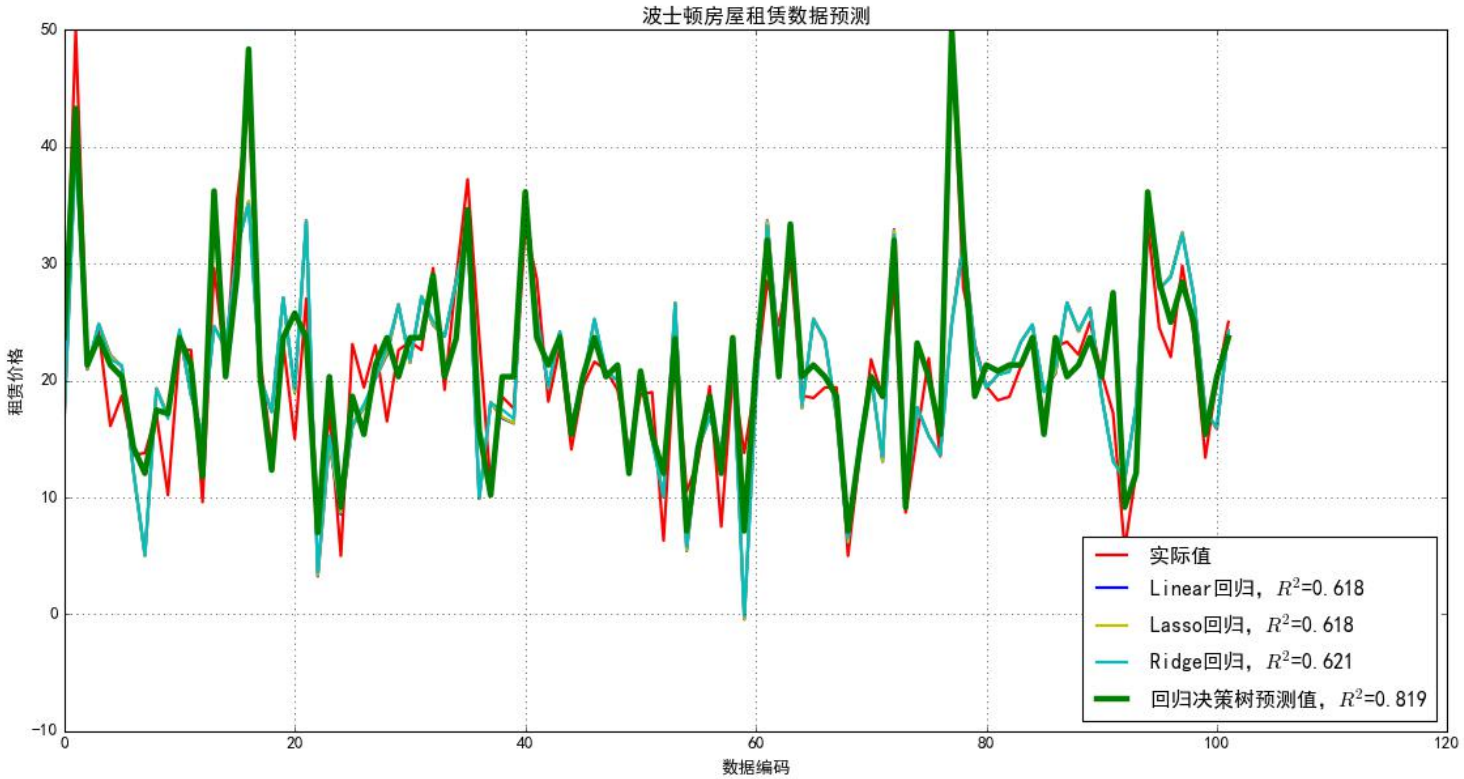
Data Set Characteristics:	Multivariate	**Number of Instances:**	506	**Area:**	N/A
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	14	**Date Donated**	1993-07-07
Associated Tasks:	Regression	**Missing Values?**	No	**Number of Web Hits:**	328263

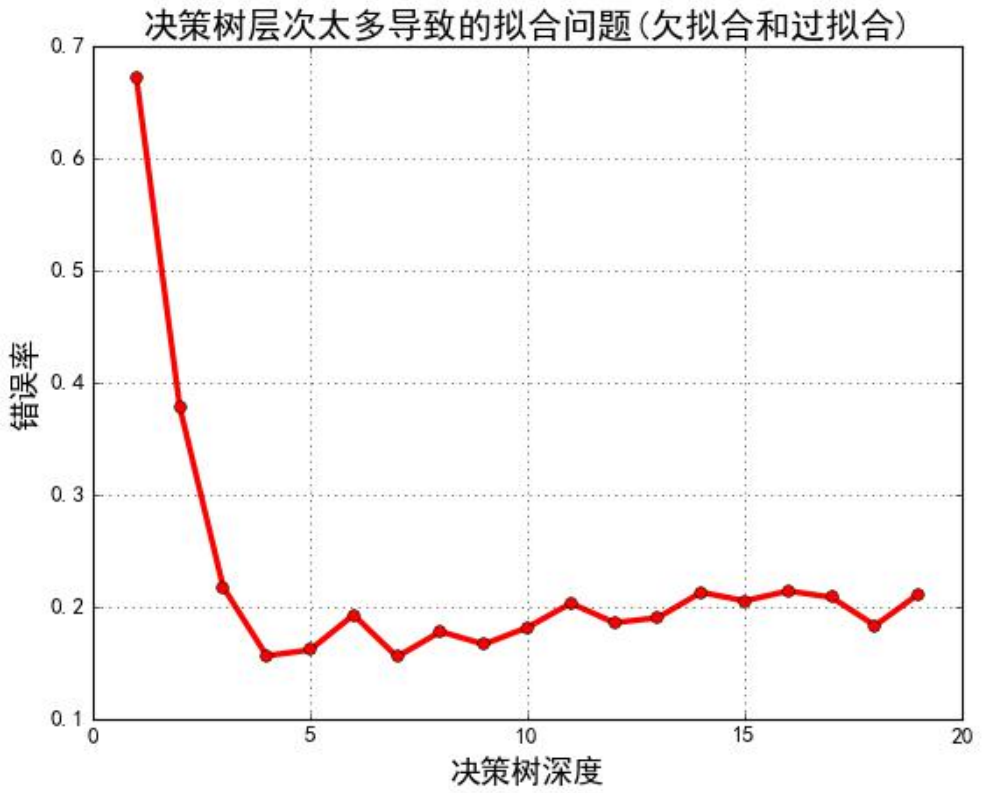
Attribute Information:

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

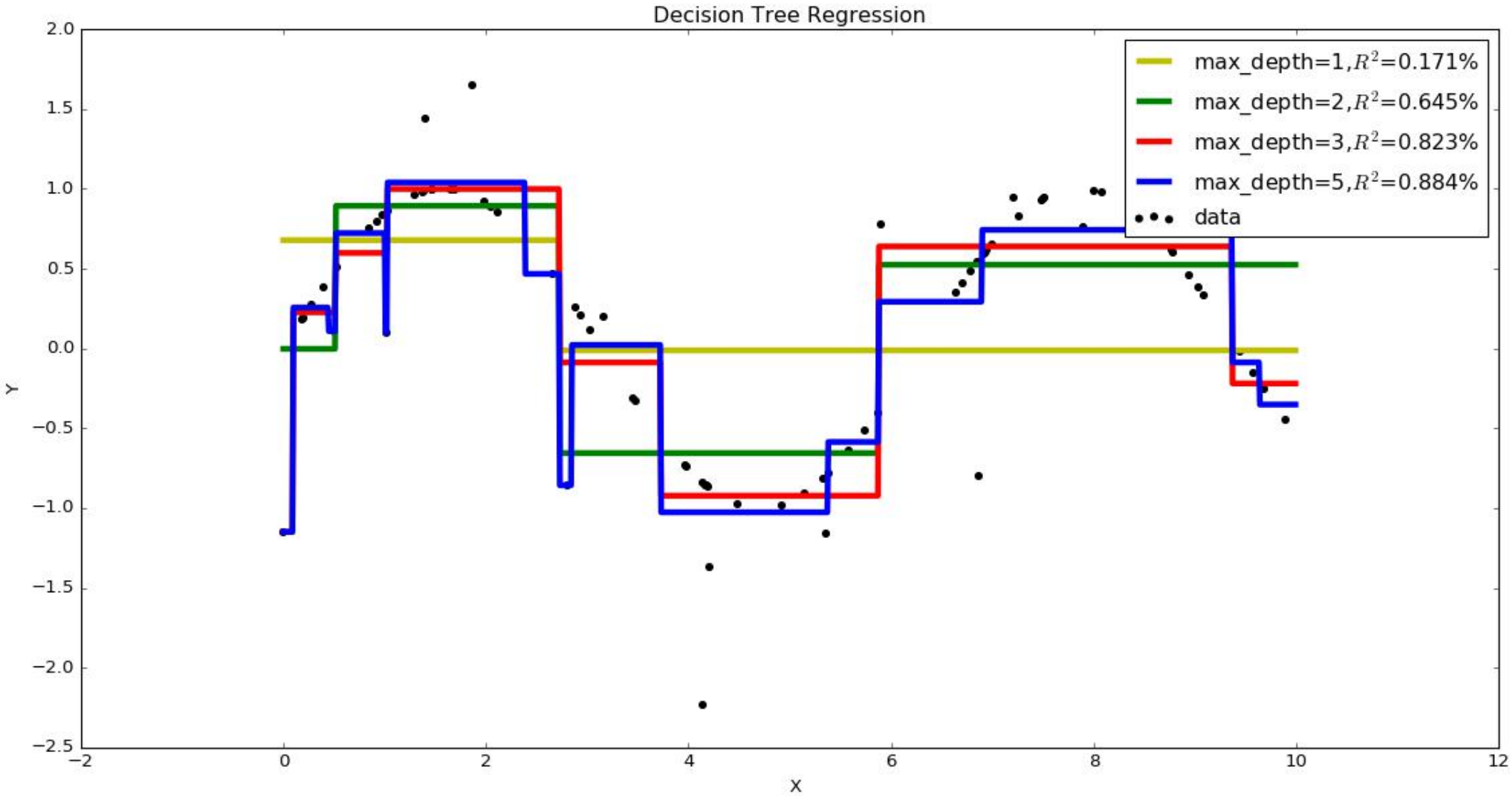
class sklearn.tree.**DecisionTreeRegressor**(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None*) ¶ [source]

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
Y
X
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。
- 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。
- 后剪枝总体思路(交叉验证)：
  - 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）

决策树剪枝过程(后置剪枝)

- 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，将其子节点进行删除操作，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
  - 重复上述操作，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：

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

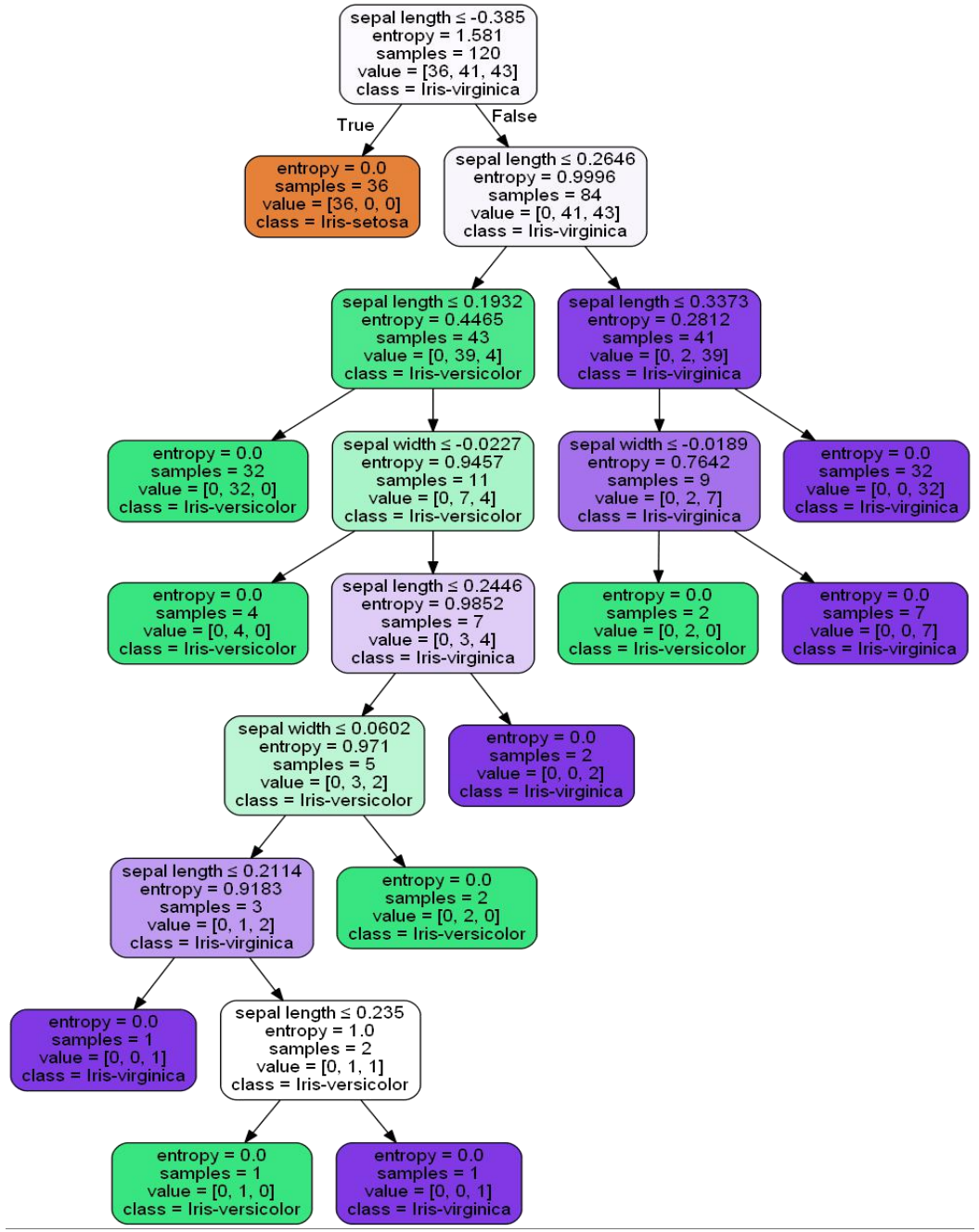
决策树剪枝损失函数及剪枝系数

- 原始损失函数 $$loss = \sum_{t=1}^{leaf} \frac{|D_t|}{|D|} \mathrm{H}(t)$$
- 叶节点越多，决策树越复杂，损失越大；修正添加剪枝系数，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

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

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

决策树可视化


sepal length ≤ -0.385
entropy = 1.581
samples = 120
value = [36, 41, 43]
class = Iris-virginica
True
False
entropy = 0.0
samples = 36
value = [36, 0, 0]
class = Iris-setosa
sepal length ≤ 0.2646
entropy = 0.9996
samples = 84
value = [0, 41, 43]
class = Iris-virginica
sepal length ≤ 0.1932
entropy = 0.4465
samples = 43
value = [0, 39, 4]
class = Iris-versicolor
sepal length ≤ 0.3373
entropy = 0.2812
samples = 41
value = [0, 2, 39]
class = Iris-virginica
entropy = 0.0
samples = 32
value = [0, 32, 0]
class = Iris-versicolor
sepal width ≤ -0.0227
entropy = 0.9457
samples = 11
value = [0, 7, 4]
class = Iris-versicolor
sepal width ≤ -0.0189
entropy = 0.7642
samples = 9
value = [0, 2, 7]
class = Iris-virginica
entropy = 0.0
samples = 32
value = [0, 0, 32]
class = Iris-virginica
entropy = 0.0
samples = 4
value = [0, 4, 0]
class = Iris-versicolor
sepal length ≤ 0.2446
entropy = 0.9852
samples = 7
value = [0, 3, 4]
class = Iris-virginica
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
entropy = 0.0
samples = 7
value = [0, 0, 7]
class = Iris-virginica
sepal width ≤ 0.0602
entropy = 0.971
samples = 5
value = [0, 3, 2]
class = Iris-versicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = Iris-virginica
sepal length ≤ 0.2114
entropy = 0.9183
samples = 3
value = [0, 1, 2]
class = Iris-virginica
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
sepal length ≤ 0.235
entropy = 1.0
samples = 2
value = [0, 1, 1]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 1, 0]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica

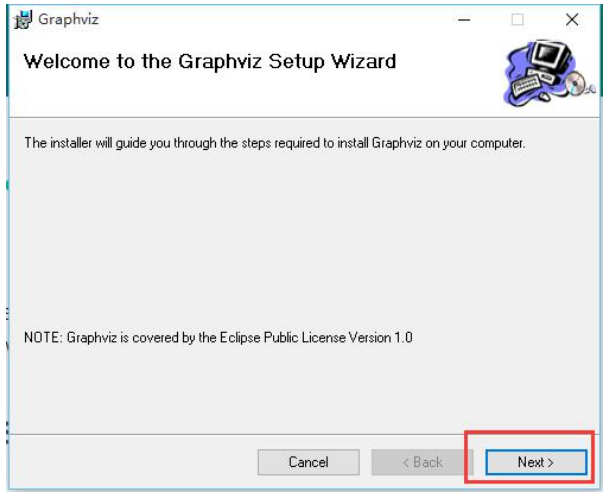
决策树可视化

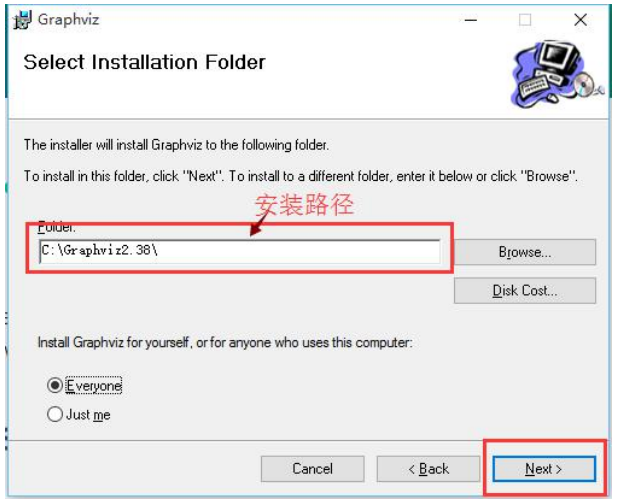
- 决策树可视化可以方便我们直观的观察所构建的树模型；决策树可视化依赖graphviz服务，所以我们在进行可视化之前，安装对应的服务；操作如下：
  - 安装graphviz服务
  - 安装python的graphviz插件/Python库： pip install graphviz
  - 安装python的pydotplus插件/Python库： pip install pydotplus

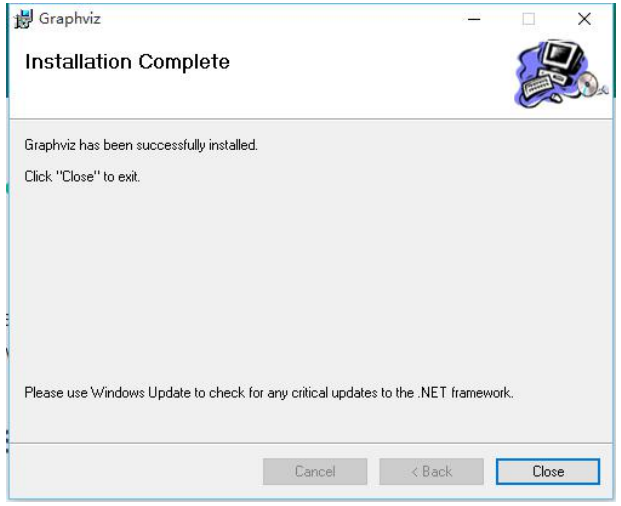
决策树可视化

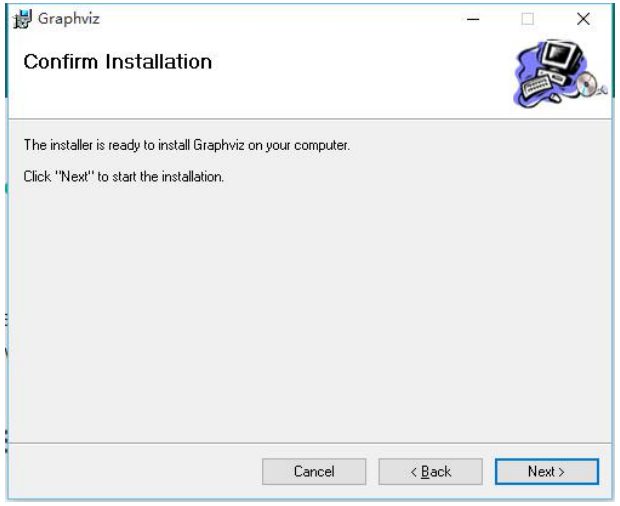
- graphviz服务安装(windows的安装)：
  - 下载安装包(msi安装包): http://www.graphviz.org/；
  - 执行下载好的安装包(双击msi安装包)；
  - 将graphviz的根目录下的bin文件夹路径添加到PATH环境变量中；

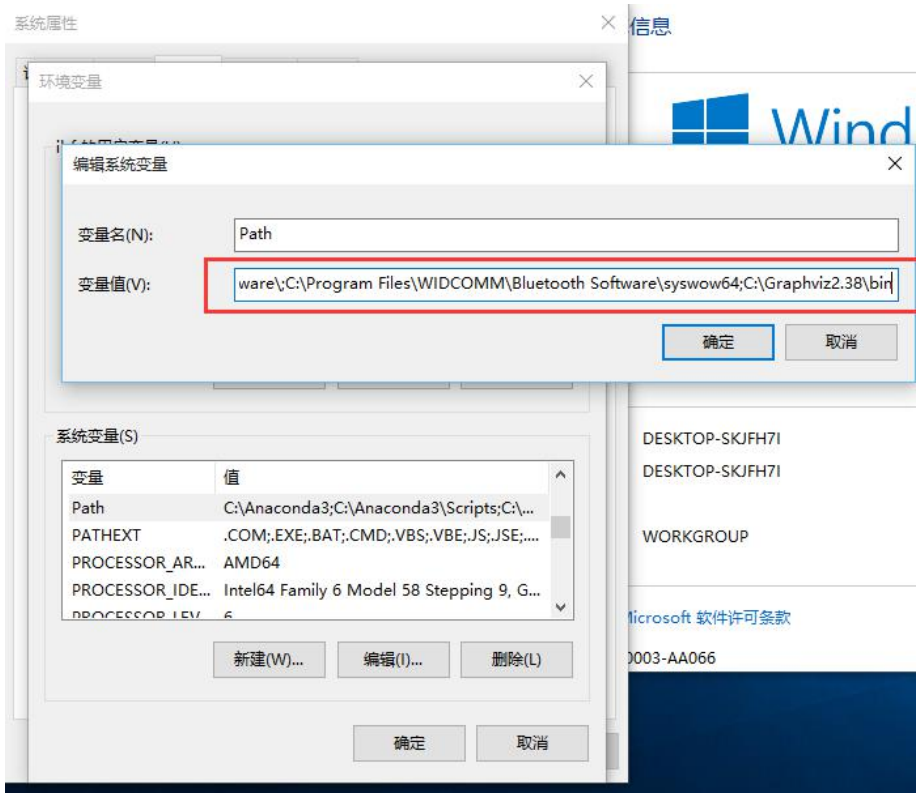
graphviz安装


Graphviz
Welcome to the Graphviz Setup Wizard
The installer will guide you through the steps required to install Graphviz on your computer.
NOTE: Graphviz is covered by the Eclipse Public License Version 1.0
Cancel
< Back
Next >


Graphviz
Select Installation Folder
The installer will install Graphviz to the following folder.
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".
安装路径
Folder:
C:\Graphviz2.38\
Browse...
Disk Cost...
Install Graphviz for yourself, or for anyone who uses this computer:
Everyone
Just me
Cancel
< Back
Next >


Graphviz
Installation Complete
Graphviz has been successfully installed.
Click "Close" to exit.
Please use Windows Update to check for any critical updates to the .NET framework.
Cancel
< Back
Close

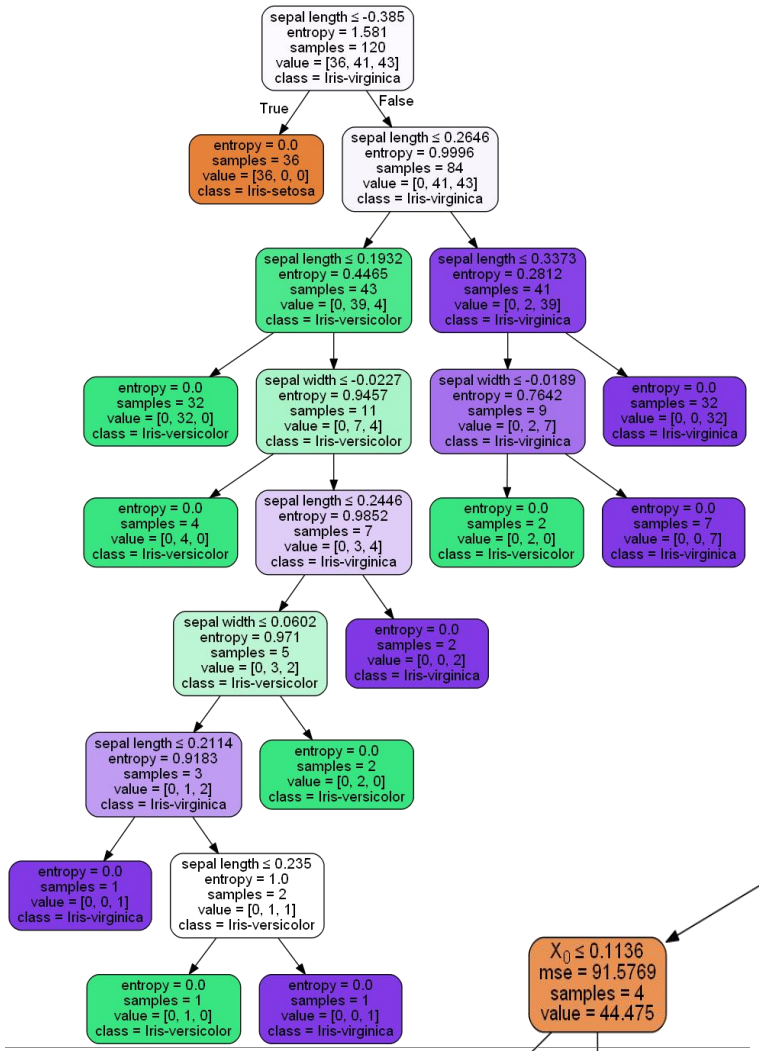

Graphviz
Confirm Installation
The installer is ready to install Graphviz on your computer.
Click "Next" to start the installation.
Cancel
< Back
Next >

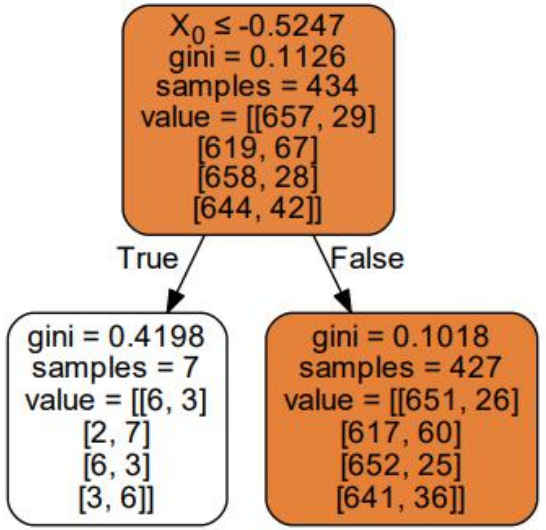

系统属性
环境变量
编辑系统变量
变量名(N):
Path
变量值(V):
ware\;C:\Program Files\WIDCOMM\Bluetooth Software\syswow64;C:\Graphviz2.38\bin
确定
取消
系统变量(S)
变量
值
Path
C:\Anaconda3;C:\Anaconda3\Scripts;C:\...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;....
PROCESSOR_AR...
AMD64
PROCESSOR_IDE...
Intel64 Family 6 Model 58 Stepping 9, G...
新建(W)...
编辑(I)...
删除(L)
确定
取消
信息
DESKTOP-SKJFH7I
DESKTOP-SKJFH7I
WORKGROUP
Microsoft 软件许可条款
0003-AA066

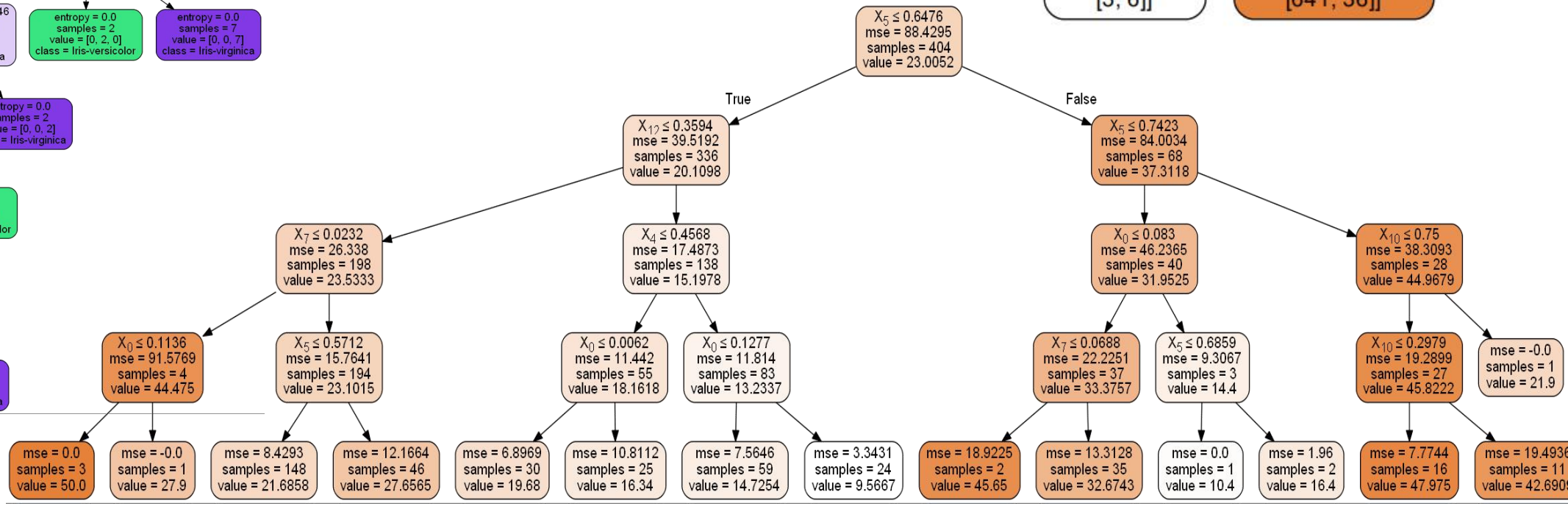
决策树可视化案例

- 方式一：将模型输出dot文件，然后使用graphviz的命令将dot文件转换为pdf格式的文件
- 方式二：直接使用pydotplus插件直接生成pdf文件进行保存
- 方式三：使用Image对象直接显示pydotplus生成的图片

决策树可视化案例


sepal length ≤ -0.385
entropy = 1.581
samples = 120
value = [36, 41, 43]
class = Iris-virginica
True
False
entropy = 0.0
samples = 36
value = [36, 0, 0]
class = Iris-setosa
sepal length ≤ 0.2646
entropy = 0.9996
samples = 84
value = [0, 41, 43]
class = Iris-virginica
sepal length ≤ 0.1932
entropy = 0.4465
samples = 43
value = [0, 39, 4]
class = Iris-versicolor
sepal length ≤ 0.3373
entropy = 0.2812
samples = 41
value = [0, 2, 39]
class = Iris-virginica
entropy = 0.0
samples = 32
value = [0, 32, 0]
class = Iris-versicolor
sepal width ≤ -0.0227
entropy = 0.9457
samples = 11
value = [0, 7, 4]
class = Iris-versicolor
sepal width ≤ -0.0189
entropy = 0.7642
samples = 9
value = [0, 2, 7]
class = Iris-virginica
entropy = 0.0
samples = 32
value = [0, 0, 32]
class = Iris-virginica
entropy = 0.0
samples = 4
value = [0, 4, 0]
class = Iris-versicolor
sepal length ≤ 0.2446
entropy = 0.9852
samples = 7
value = [0, 3, 4]
class = Iris-virginica
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
entropy = 0.0
samples = 7
value = [0, 0, 7]
class = Iris-virginica
sepal width ≤ 0.0602
entropy = 0.971
samples = 5
value = [0, 3, 2]
class = Iris-versicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = Iris-virginica
sepal length ≤ 0.2114
entropy = 0.9183
samples = 3
value = [0, 1, 2]
class = Iris-virginica
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
sepal length ≤ 0.235
entropy = 1.0
samples = 2
value = [0, 1, 1]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 1, 0]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica


X0 ≤ -0.5247
gini = 0.1126
samples = 434
value = [[657, 29]
[619, 67]
[658, 28]
[644, 42]]
True
False
gini = 0.4198
samples = 7
value = [[6, 3]
[2, 7]
[6, 3]
[3, 6]]
gini = 0.1018
samples = 427
value = [[651, 26]
[617, 60]
[652, 25]
[641, 36]]


X5 ≤ 0.6476
mse = 88.4295
samples = 404
value = 23.0052
True
False
X12 ≤ 0.3594
mse = 39.5192
samples = 336
value = 20.1098
X5 ≤ 0.7423
mse = 84.0034
samples = 68
value = 37.3118
X7 ≤ 0.0232
mse = 26.338
samples = 198
value = 23.5333
X4 ≤ 0.4568
mse = 17.4873
samples = 138
value = 15.1978
X0 ≤ 0.083
mse = 46.2365
samples = 40
value = 31.9525
X10 ≤ 0.75
mse = 38.3093
samples = 28
value = 44.9679
X0 ≤ 0.1136
mse = 91.5769
samples = 4
value = 44.475
X5 ≤ 0.5712
mse = 15.7641
samples = 194
value = 23.1015
X0 ≤ 0.0062
mse = 11.442
samples = 55
value = 18.1618
X0 ≤ 0.1277
mse = 11.814
samples = 83
value = 13.2337
X7 ≤ 0.0688
mse = 22.2251
samples = 37
value = 33.3757
X5 ≤ 0.6859
mse = 9.3067
samples = 3
value = 14.4
X10 ≤ 0.2979
mse = 19.2899
samples = 27
value = 45.8222
mse = -0.0
samples = 1
value = 21.9
mse = 0.0
samples = 3
value = 50.0
mse = -0.0
samples = 1
value = 27.9
mse = 8.4293
samples = 148
value = 21.6858
mse = 12.1664
samples = 46
value = 27.6565
mse = 6.8969
samples = 30
value = 19.68
mse = 10.8112
samples = 25
value = 16.34
mse = 7.5646
samples = 59
value = 14.7254
mse = 3.3431
samples = 24
value = 9.5667
mse = 18.9225
samples = 2
value = 45.65
mse = 13.3128
samples = 35
value = 32.6743
mse = 0.0
samples = 1
value = 10.4
mse = 1.96
samples = 2
value = 16.4
mse = 7.7744
samples = 16
value = 47.975

决策树可视化案例

```
# 方式一：输出形成dot文件，然后使用graphviz的dot命令将dot文件转换为pdf
from sklearn import tree
with open('iris.dot', 'w') as f:
    f = tree.export_graphviz(model, out_file=f)
# 命令行执行dot命令：  dot -Tpdf iris.dot -o iris.pdf
```

```
# 方式二：直接使用pydotplus插件生成pdf文件
from sklearn import tree
import pydotplus
dot_data = tree.export_graphviz(model, out_file=None)
graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_pdf("iris2.pdf")
# graph.write_png("d.png")
```

```
# 方式三：直接生成图片
from sklearn import tree
from IPython.display import Image
import pydotplus
dot_data = tree.export_graphviz(model, out_file=None,
                                feature_names=['sepal length', 'sepal width', 'petal length', 'petal width'],
                                class_names=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
                                filled=True, rounded=True,
                                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```


北风网
IBEIFENG.COM