分类模型-3：决策分类树 and 集成模型
Decision Tree & Ensemble Learning

决策树、装袋法、随机森林、提升法

1

内容简介

- 决策树：启发示例
- 决策树模型
  - 主要类别
  - 基本原理
  - 决策树在R中实现示范
  - 案例
- 基于树的集成模型
  - 装袋法（Bagging，bootstrap aggregation）
  - 随机森林（Random Foreast）
  - 提升法/助推法（Boosting）
  - 随机森林和助推法在R中实现示范

2

内容简介 – 决策树 – 启发示例

- 目的
- 示例描述
- 解决思路：
  - 用决策树映射特征和结果的关系
  - 需解决三个问题
  - 用信息熵衡量不确定性，用信息增益衡量决策规则价值；
  - 三个问题的解决

3

目的

- 理解机器如何根据三个简单计算规则构建决策树；
- 试着根据案例进行类比，体会如何把更多的管理决策问题转化为树来求解。

4

示例描述

编号	特征/属性				标签
	天气	温度	湿度	风否	
1	晴	热	大	无	否
2	晴	热	大	有	否
3	多云	热	大	无	是
4	雨	中	大	无	是
5	雨	冷	正常	无	是
6	雨	冷	正常	有	否
7	多云	冷	正常	有	是

5

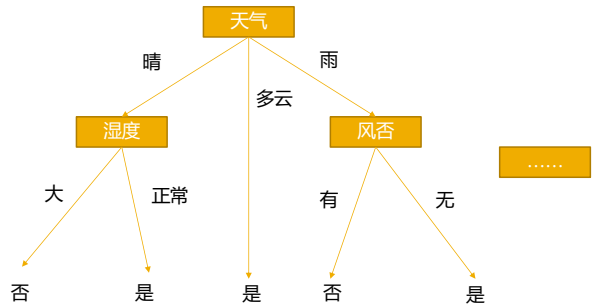
示例-数据-续

编号	特征/属性				标签
	天气	温度	湿度	风否	
8	晴	中	大	无	否
9	晴	冷	正常	无	是
10	雨	中	正常	无	是
11	晴	中	正常	有	是
12	多云	中	大	有	是
13	多云	热	正常	无	是
14	雨	中	大	有	否

6

用决策树映射特征和结果的关系-示意

- 节点、分叉、叶子



7

需解决三个问题

- （1）从哪个特征开始分割？
- （2）后续节点选择哪些特征？
- （3）何时可以停止树的构建？

8

度量

- 用信息熵-entropy 度量不确定性，用信息增益-information gain 度量决策规则价值；
- 在管理决策中，信息的基本作用是消除决策的不确定性，决策时掌握信息越多就越可能做出理智决策；
- 信息熵-entropy（香农-Shannon熵）：
  - 意外性/信息的数学期望，用于度量随机变量的不确定性/信息量；
  - 定义：$H(X)=E_{p(x)}\left(\log\frac{1}{P(X=x_i)}\right)=-\sum_{x_i\in X}P(X=x_i)\log P(X=x_i)$
  - 性质：
    - $H(X)\geq 0$;
    - $P(X=x_i)=0$或1时(完全确定),定义$H(X)=0$,与$\lim_{p\to 0^+}(-p\cdot\log p)=0$一致；
    - 当$P(X=x_i)=1/C$ 时，变量的不确定度最大，熵最大;
  - 不确定性越高，熵值越大

9

度量

- 用信息熵-entropy 度量不确定性，用信息增益-information gain 度量决策规则价值；
  - 观察到某人对一件事的16次选择，8次选择是，8次选择否；则其选择的不确定性/信息熵可以计算如下：
    $$-\frac{8}{16}\times\log_2\frac{8}{16}-\frac{8}{16}\times\log_2\frac{8}{16}=1$$
  - 若16次选择，4次选择是，12次选择否；则其选择的不确定性/信息熵可以计算如下：
    $$-\frac{4}{16}\times\log_2\frac{4}{16}-\frac{12}{16}\times\log_2\frac{12}{16}=0.811$$
  - 若16次选择，16次选择是；则其选择的不确定性/信息熵0：
    $P(X=x_i)=0$或1时(完全确定)，定义$H(X)=0$

10

构造决策树的基本思想

- 随着树深度的增加，节点的熵迅速降低；
- 信息增益 - information gain：用于度量信息熵减少的程度；用某决策规则（分叉）/特征使用前后两次信息熵差值表示，差值越大说明该决策规则价值越大（不确定性减少得越多）；

天气	是	否	温度	是	否		
晴	2	3	热	2	2		
多云	4	0	中	4	2		
雨	3	2	冷	3	1		
湿度	**是**	**否**	**风否**	**是**	**否**	**是**	**否**
大	3	4	无	6	2	9	5
正常	6	1	正常	3	3		

11

三个问题的解决

- （1）从哪个特征开始分割？（根节点）
  - 选择信息增益最大的特征作为根节点；
  - 不使用任何特征时，熵=$-\frac{5}{14}\times\log_2\frac{5}{14}-\frac{9}{14}\times\log_2\frac{9}{14}=0.94$
  - 若用"天气"为根节点，则使用该属性规则后的信息熵：
    - 天气=晴 时，"否"的概率$\frac{3}{5}$，"是"的概率$\frac{2}{5}$，熵：
      - $-\frac{3}{5}\times\log_2\frac{3}{5}-\frac{2}{5}\times\log_2\frac{2}{5}=0.971$
    - 天气=雨 时，"否"的概率$\frac{2}{5}$，"是"的概率$\frac{3}{5}$，熵：
      - $-\frac{2}{5}\times\log_2\frac{2}{5}-\frac{3}{5}\times\log_2\frac{3}{5}=0.971$
    - 天气=多云 时，"否"的概率 0，"是"的概率 1，熵=0
    - 天气=晴 的概率$\frac{5}{14}$，天气=雨 的概率$\frac{5}{14}$，天气=多云 的概率$\frac{4}{14}$，
      - 天气的信息熵 = $\frac{5}{14}\times0.971+\frac{5}{14}\times0.971+\frac{4}{14}\times0=0.693$
    - 使用天气为决策规则对象的信息熵为0.693
  - 使用天气特征作为根节点的信息增益为：
    - $0.94-0.693=0.247$

12

三个问题的解决

- （1）从哪个特征开始分割？（根节点）--续
  - 同理，分别使用温度、湿度、风否作为根节点的信息增益分别为0.029、0.152、0.048；
  - 所以选择**天气**为根节点特征（因为它能让信息熵下降最快）
- （2）后续节点选择哪些特征？
  - 根据信息增益计算结果而定；
  - 例如，天气=晴 这一分叉（**条件**）下，根据Data，再分别计算温度、湿度、风否作为后续节点的信息增益，其中增益最大的作为后续节点；其余类推；
- （3）何时可以停止树的构建？
  - 信息增益为0时。

13

注意

- 借助信息熵、信息增益构建决策树的算法史称C4.5/C5.0算法；
- 此外，还有基于基尼系数、卡方检验等构建决策树的算法。

14

启发示例 - 小结

- 理解机器如何根据三个简单计算规则构建决策树；
- 试着根据案例进行类比，体会如何把更多的管理决策问题转化为树来求解。

15

内容简介

- 决策树：启发示例
- 决策树模型
  - 主要类别
  - 基本原理
  - 决策树在R中实现示范
  - 案例
- 基于树的集成模型
  - 装袋法（Bagging，bootstrap aggregation）
  - 随机森林（Random Foreast）
  - 助推法（Boosting），/提升法
  - 随机森林和助推法在R中实现示范

16

决策树 -

- 场景1：医生诊断流感
  - 向患者了解症状：头痛、发热、流涕等；
  - 医技检查：量体温、查验血等；
  - 做诊断
- 场景2：银行信用卡评估（简化版本哈）
  - 借贷方收入高，违约风险低；
  - 借贷方收入中等且教育程度本科或研究生，则不易违约；
  - 借贷方收入中等但高中或以下学历，易违约；
  - 借贷方收入低，易违约。
- 在数据科学领域，此类用树形式表示的模型称为决策树模型，为经典模型之一；可转化为**IF-Then**形式的决策规则，符合人类的决策思维习惯；
  - 专家系统（Expert System）：人工智能早期，由人类专家制定规则并构造复杂的树，帮助人们进行分类/决策，但这样的树的构建**不依赖于数据**。这不是数据科学意义上的决策树。
- **强可解释性**且能集成令其应用广泛，例如金融、医疗保健等领域。

24

决策树 – decision tree

- **决策树**
  - 主要类别
  - 基本原理
  - 实现
  - 案例
- 注意：本章主要讨论分类问题（**分类树**），用于回归问题的**回归树**和**模型树**参见“回归模型”

27

决策树 – decision tree

- 一类基于树的回归和分类方法：
  - 主要类别：回归树-regression tree、分类树-classification tree
  - 训练：主要根据分层、分割的方式将预测变量空间划分为一系列简单区域；
  - 预测：对待预测观测/实例，根据其所属区域的响应变量的平均值（回归问题）或众数（分类问题）进行预测；
  - 划分预测变量空间的分裂规则可表示为一棵树，所以称为决策树。
- 基于树的**集成**方法：
  - 简单的决策树方法简便且易于解释，但预测准确性通常低于其他经典预测方法；（容易过拟合）
  - 装袋法(bagging)、随机森林(random forest)、提升法(boosting)；
  - 先构建多棵树，再综合这些树，最后根据表决产生预测；这会极大提升预测的准确性（虽然会损失一些可解释性）。

28

决策树

- 基本原理：通过对特征空间分区/划分来预测；
  - 两步骤：
    - （1）将特征空间（即$X_1, X_2, \ldots, X_p$的可能取值构成的集合）分割成J个互不重叠的区域$R_1, R_2, \ldots, R_J$；
    - （2）对落入区域R_j的每一个待预测观测/实例，令其预测值等于R_j上训练集的响应值的平均值（回归树）或众数（分类树）。
  - 关键：步骤（1）如何划分区域$R_1, R_2, \ldots, R_J$？
    - 区域形状：理论上，可以是任意的；实践中，将特征空间划分为高维矩形（简化模型、增强可解释性）；
    - 划分目标：回归问题，常用MSE来度量精度；分类问题，常用误分率度量；（尽量让一个分支区域中的实例/观测属于同一类别）
    - 划分策略：以min(划分目标)为目标，理想中，需要考虑将特征空间划分为J个区域的所有可能；实践中，因为计算上不可行，通常采用**自上而下、贪婪**的方法：例如递归二叉分裂。

29

实现

- 递归二叉分裂
  - 对于当前待分割区域，选择**预测变量X_j和分割点/值s_j**，将该区域分为2个区域$R_1(j,s)=\{X|X_j\le s_j\}$和$R_2(j,s)=\{X|X_j>s_j\}$，使得$\sum_{i:x_i\in R_1} obj_i+\sum_{i:x_i\in R_2} obj_i$尽可能小。即：
    $$\min_{j,s_j}(\sum_{i:x_i\in R_1} obj_i+\sum_{i:x_i\in R_2} obj_i) \rightarrow j, s_j$$
  - 重复上述步骤，继续以min为目标寻找分割数据集的"局部"最优预测变量和最优分割点直到符合某个停止准则：例如决策树达到规定大小（过于复杂易过拟合）；
  - 区域$R_1, R_2, \dots, R_J$产生后，可利用其预测。

30

分类树-分类效果度量指标

- **误分率**:
  - 要将给定区域内的观测值分到此区域训练集上最常出现的类中，则分类错误率可定义为：该区域R_j的训练集中**非最常见类**所占的比例：
    $$E=1-\max_c \hat{p}_{jc}$$
    $\hat{p}_{jc}$：表示区域R_j的训练集中第c类所占的比例；
  - 但是，可以证实误分率对树规模的增长不敏感；
- **纯度**：尽量让一个分支区域中的实例/观测属于同一类别
  - 基尼系数（Gini index）
    $$G=\sum_{c=1}^{C}\hat{p}_{jc}(1-\hat{p}_{jc})$$
  - 熵（entropy）[C5.0算法使用分割前后的**信息增益**最大作分割标准]
    $$\text{Entropy}(S)=-\sum_{c=1}^{C}\hat{p}_{jc}\log_2\hat{p}_{jc}$$
  - 基尼系数和熵在数值上相当接近；若第j个区域纯度较高，则熵值较小。

31

信息增益

- 作为决策树特征变量择取依据的信息增益；
- **C5.0算法**使用分割前后的**信息增益**最大作分割标准；
- 在决策树上进行特征选择时，总是优先选择互信息$I(Y;X_i)$最大（亦即信息增益$IG(Y;X_i)$最大）的特征变量X_i，因为X_i能为识别Y带来最大的信息量。

32

常见的决策树算法

- Ross Quinlan 的 ID3算法开创了决策树算法先河，后期提出的对**ID3**改进算法：**C4.5(J48)→C5.0**一脉相承；
  - ID3：预测变量（离散）、响应变量（离散）、信息增益
  - ID4.5：预测（离散、连续）、响应（离散）、信息增益率
  - library(C50)
- Breiman和Friedman等提出**CART**算法同时处理分类和回归问题；
  - 预测（离散、连续）、响应（离散、连续）、Gini和方差
  - library(**rpart**)

33

分类树-分类效果提升

- 剪枝：
  - 减少过拟合，便于推广到对未知数据的预测；
  - 分类：
    - 预剪枝：决策树达到的决策或区域仅含少量实例时；能减少工作量，但不知道是否会错过细微但重要的模式；
    - 后剪枝：生长得太大时再修剪到合适的大小；
  - C5.0算法能自动剪枝；从而在过拟合和欠拟合间达到平衡。

34

分类树-分类

- 对不同类别实例误分的代价不对称的情形：
  - 信贷风险评估：把高风险的客户误分为低风险，可导致重大损失；
  - 医技检查中：假阴性比假阳性带来更大的风险；
- 对策：
  - 引入**惩罚因子**，在C50::C5.0(train, class, trials=1,costs=NULL)中通过代价矩阵costs实现；
  - 例子参见教材P99-100

35

决策树 – decision tree

- **预测/决策：**从根节点开始，测试待分类实例中相应特征，并按其值选择输出分支，直到某叶节点，最后将叶节点存放的类别作为决策结果;
- **优点：**
  - 原理简单，决策过程直观，易被人类理解（可解释性强）；
  - 能处理连续预测变量和分类预测变量；易于可视化；
  - 不需要任何假设（如回归）、领域知识或参数设置；
- **缺点**：
  - 采用贪婪算法，易于陷入局部最优；模型不稳定易受噪音影响；
  - **单棵决策树**，确定决策边界时每次只涉及单个变量的逻辑判断，导致决策边界是平行于坐标轴的直线，这限制了对分类边界的表达，准确性较差，**非常容易过拟合，很少独立使用**。→ 集成模型
- **集成应用：**已成功用于医、制造、生物以及商业等诸多领域。

36

R语言中决策树的实现

- C50包中的C5.0()函数
  - C5.0(credit_train[-17], credit_train$default)
  - 可以引入代价矩阵-cost matrix 来校正权重的默认设置
    C5.0(credit_train[-17], credit_train$default, costs = error_cost)
- rpart包中的rpart()函数
  - rpart(default~., data=credit.train, method="class", parms=list(split="information"))
  - rpart包实现了Category And Regression Tree-CART-分类回归树；
  - rpart.plot包中rpart.plot(model.rpart)实现了决策树的可视化；

37

决策树模型 - 常用超参

- **minsplit** - 拆分节点所需的最小观测数；若一个节点中观测的数量少于指定的数量，则该节点将不会被进一步拆分；
- **maxdepth** - rpart中树的最大深度 ；若一个节点已在该深度，它就不会被进一步拆分；
- **cp** - 为树各深度计算复杂性参数(rpart 中的 cp - complexity parameter)；若某个深度的 cp 值小于给定阈值，则该级别的节点将不会进一步分裂。换言之，若向树添加深一层不会提高模型的性能-cp，则不要拆分节点；
- **minbucket** – rpart中指叶节点中的最少观测数；若拆分一个节点会导致叶节点包含的观测少于 minbucket，则该节点不会被拆分。

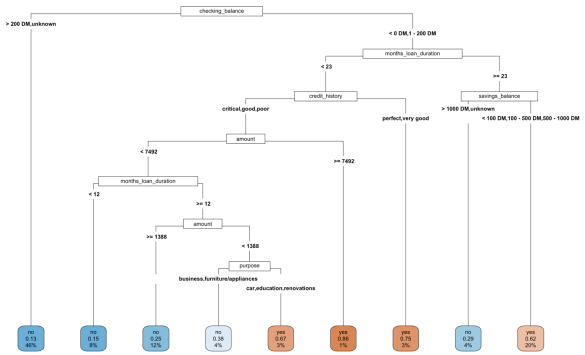
38

决策树 – mlr代码示范

```
task <- makeClassifTask(data = myData, target = "myType")
treeLearner <- makeLearner("classif.rpart"); getParamSet(treeLearner)     # install.packages("rpart")
treeParamSpace <- makeParamSet( makeIntegerParam("minsplit", lower = 5, upper = 20),
                                makeIntegerParam("minbucket", lower = 3, upper = 10),
                                makeNumericParam("cp", lower = 0.01, upper = 0.1),
                                makeIntegerParam("maxdepth", lower = 3, upper = 10))
randSearch <- makeTuneControlRandom(maxit = 200)
cvForTuning <- makeResampleDesc("CV", iters = 5)

library(parallel); library(parallelMap)
parallelStartSocket(cpus = detectCores())
tunedTreePars <- tuneParams(treeLearner, task = task,
                            resampling = cvForTuning,
                            par.set = treeParamSpace, control = randSearch)
parallelStop()

tunedTree <- setHyperPars(treeLearner, par.vals = tunedTreePars$x)
tunedTreeModel <- train(tunedTree, task)

library(rpart.plot)  # install.packages("rpart.plot")
treeModelData <- getLearnerModel(tunedTreeModel)
rpart.plot(treeModelData, roundint = FALSE, box.palette = "auto", type = 5)
```


39

rpart.plot()函数绘制的决策树图-示例



40

决策树 – decision tree – 教材案例

- **案例：**
  - 信贷风险管理：个人信用风险评估；
  - 数据来源：UCI机器学习数据仓库-credit.csv；
    - http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29
  - 背景：
    - 许多银行对申请者的贷款/信用卡申请被拒绝/批准给出明确的理由，自动化的信用评分模型能提供辅助决策；
    - 小额信贷服务提供方实时自动信用评分成为可能；
  - 实现代码：
    - chap5-DecisionTree-Updated.R
    - 含数据探索、准备、训练、预测、可视化、提升等；

41

决策树 – decision tree

- 启发示例
- 决策树
  - 主要类别
  - 基本原理
  - 决策树在R中实现示范
  - 教材案例
- **集成模型**
  - 装袋法（Bagging，bootstrap aggregation）
  - 随机森林（Random Foreast）
  - 助推法（Boosting），/提升法
  - 随机森林和助推法在R中实现示范

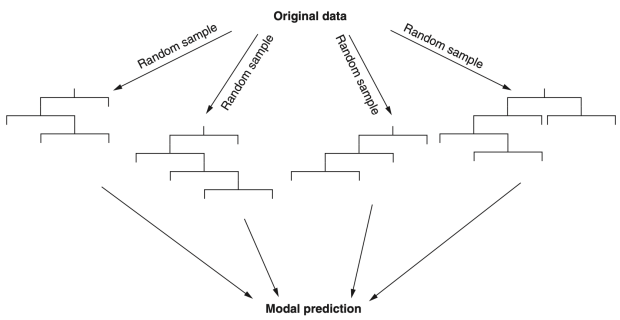
42

装袋法（Bagging，bootstrap aggregation）

- 是一种利用Bootstrap（有放回随机抽样）的通用方法，可用于利用任何回归或分类模型来构建集成模型；
- 装袋法Bagging - 算法：
  - For 1 to m：
  - 从原始数据集生成Bootstrap样本；
  - 在生成样本上训练未修剪的树；
  - 利用构建的树预测连续变量值或分类；
  - 集成m棵树的结果（平均或投票）
- 装袋法可有效降低单个模型预测的方差、能改进不稳定模型（如决策树）的预测性能；但通常计算成本增大（好在容易并行化）、解释能力比未装袋模型要弱很多（例如装袋树无法得到单棵树那样简洁的IF-Then规则）

43

装袋法 - bootstrap aggregation



44

装袋法（Bagging，bootstrap aggregation）

- 为什么装袋法性能比单模型优越：
  - 抽样：每次Bootstrap抽样时，因为是有放回抽样，所以更可能选择靠近分布中心的数据，而不是靠近分布极端的数据（异常值、噪音）；
  - 预测：一些 Bootstrap 样本可能包含较多极端值，在此上训练的模型会做出糟糕的预测，但装袋法最终会用集成模型中各模型预测的汇总（例如投票）作为最终的预测。

45

随机森林（Random Foreast）

- RF与装袋法相比对单棵树的预测方差进一步降低
  - 随机抽取数据-行（如装袋法Bootstrap）在其上建模；
  - 随机抽取变量-列（在分裂点上）在其上选择节点变量；
- 随机森林的基本算法：

选择模型数量m；
For 1 to m：
从原始数据集生成Bootstrap样本；
在生成样本上训练未修剪的树；
在每个分裂点上：
随机抽取 k (k < p) 个预测变量；
在 k 个变量中择优（如信息增益最大）选用划分数据的变量；
用集成树预测：集成m棵树的结果（平均或投票）

随机森林（Random Foreast）

- 与装袋树相比，随机森林预测方差更小（除了数据Bootstrap抽取随机，预测变量抽取亦随机）；随机森林计算量比装袋树更小（更少的预测变量）；随机森林通常可以避免过拟合；随机森林每棵树独立生长亦可并行；
- 随机森林调参建议：
  - 使用1000棵（ntree）树或更多（若CV显示性能还能提升）；
  - 变量数目 k （mtry）在 2:p 之间等间隔划分（k可考虑5或更大，若CV则通常可取小一些）

助推法（Boost）- 以分类为例

- 助推法中，多个弱分类器与强分类器结合，算法很多；基本思想：后续模型从前序模型的错误中学习（模型间有依赖）；
- 自助助推法(AdaBoost) 算法：

每个样本赋值相同的**样本权重**（1/n）；
For k=1 to K：
对加权后的样本Boostrap训练集，训练第k个分类器，算误分率err_k
计算第k个分类器的**模型权重**（例如 $0.5\times\ln\frac{1-err_k}{err_k}$ ）
用集成模型做预测，更新**样本权重**：增加误判样本的权重，
减小正判样本的权重；
预测：用最终集成模型做预测（结合k个模型的模型权重和预测结果：
每个模型都将单独投票，但每个投票将由模型权重进行加权）。

- 在迭代中，分类困难的样本的权重会不断加大直到算法找到能正确分类这些样本的分类器模型（强）；分类难的样本出现机会大，后续模型性能强；
- 助推法可以用于任何分类技术，但助推分类树是常用方法（因为分类树是一种低偏差/高方差技术，树的集合有助于降低方差，产生一个低方差低偏差的结果）；助推法亦可用于回归问题；

XGBoost – 极端梯度提升

- **XGB**oost (e**X**treme **G**radient **B**oosting，2014) 算法
  - 是梯度提升算法(gradient boosting)的一种，后续模型以最小化先前模型的残差为目标；
  - 梯度提升法的思想：从前序集成模型的残差中学习；（注：AdaBoost从前序模型的错误中学习）；
  - 优点：避免单个预测变量对预测结果产生过大的影响，有助于防止过度拟合；能同时减小偏差与方差。

随机森林与自助法

- 随机森林，每棵树独立构建，对最终结果贡献均等；助推法中的树依赖于之前生成的树，且不同的树对最终结果的贡献不同；
- 二者的预测性能均不错；随机森林因为每棵树独立构建，能并行，因而比自助法计算效率高。

- 注：boosting翻译自助法，又作提升法。

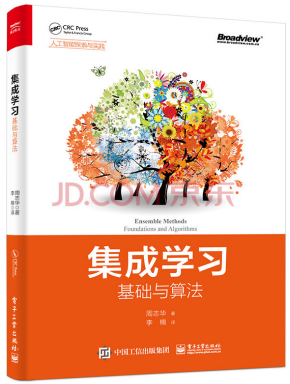
50

集成模型/算法

- stacking集成：
  - 使用不同的算法来学习子模型（bagging和boosting用相同算法）；
  - 思想：创建一些在不同特征空间区域的基模型（一般会使用彼此差别大的基模型），然后基模型作出的预测以及原始特征变量，共同作为另一个模型（stack模型）的预测变量。
  - 通常复杂且难以实现。
- 像 bagging、boosting 、stacking这样的集成方法本身并不是严格的机器学习算法，但它们是在其他机器学习模型的基础上应用的算法；
- 集成算法常用于基于树的模型，但同样可将 bagging 和 boosting 用于其他机器学习算法，如 knn 和线性回归模型。

51

参考文献推荐



52

决策树 – decision tree

- 启发示例
- 决策树
  - 主要类别
  - 基本原理
  - 决策树在R中实现示范
  - 教材案例
- **集成模型**
  - 装袋法（Bagging，bootstrap aggregation）
  - 随机森林（Random Foreast）
  - 助推法（Boosting），/提升法
  - 随机森林和助推法在R中实现示范

53

随机森林 - 重要超参

- ntree - 森林中单棵树的数量；
- mtry - 每个节点随机抽取的特征变量的数量；
- nodesize - 叶节点中允许的最小观测数；
- maxnodes——允许的最大叶节点数；

54

随机森林 – 代码示范

```
task <- makeClassifTask(data = myData, target = "myType")

learnerForest <- makeLearner("classif.randomForest")    #install.packages("randomForest")
forestParamSpace <- makeParamSet( makeIntegerParam("ntree", lower = 1000, upper = 1000),
                                  makeIntegerParam("mtry", lower = 6, upper = 12),
                                  makeIntegerParam("nodesize", lower = 1, upper = 5),
                                  makeIntegerParam("maxnodes", lower = 5, upper = 20))
randSearch <- makeTuneControlRandom(maxit = 100)
cvForTuning <- makeResampleDesc("CV", iters = 5)

parallelStartSocket(cpus = detectCores())
tunedForestPars <- tuneParams( learnerForest, task = task,
                               resampling = cvForTuning,
                               par.set = forestParamSpace, control = randSearch)
parallelStop()

tunedForest <- setHyperPars(learnerForest, par.vals = tunedForestPars$x)
tunedForestModel <- train(tunedForest, task)
```


55