R **机器学习**

第 10 讲　决策树

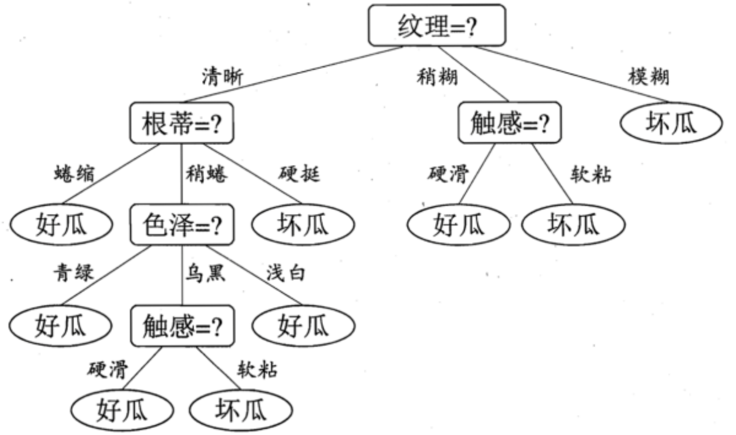
张敬信

2022 年 10 月 23 日

哈尔滨商业大学

- 决策树是经典的数据挖掘算法，简单直观、通俗易懂，不需要任何专业领域知识和复杂数学推理，既可以做分类也可以做回归，且结果具有很强的解释性。
- 决策树需要数据量可以很少，既能处理连续变量也能处理离散变量，且不需要做特征缩放，对缺失值、异常值也不敏感。
- **使用决策树进行分类的过程，可认为是用** if-then **规则基于特征对样本进行分类的过程**：从根节点开始，对实例的某一个特征进行测试，根据测试结果，将实例分配到其子结点；此时，每一个子结点对应着该特征的一个取值。如此递归向下移动，直至达到叶结点，最后将样本分配到叶结点的类中。

- 以对西瓜分类为例：



纹理=?
清晰
稍糊
模糊
根蒂=?
触感=?
坏瓜
蜷缩
稍蜷
硬挺
硬滑
软粘
好瓜
色泽=?
坏瓜
好瓜
坏瓜
青绿
乌黑
浅白
好瓜
触感=?
好瓜
硬滑
软粘
好瓜
坏瓜

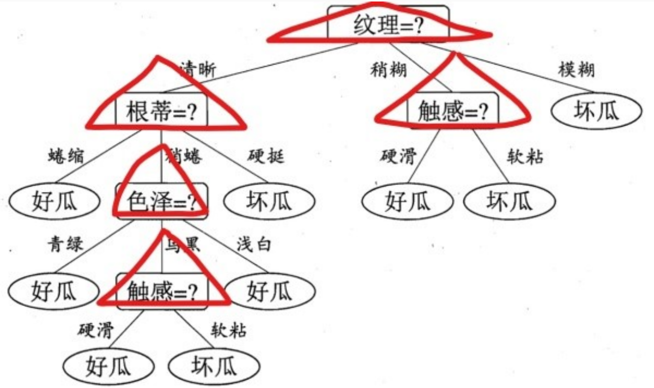
- 上图完整表达了选择一个好西瓜的策略（决策树），其中内部结点（方框）表示判断条件，叶结点（椭圆框）表示决策结果，有向边（连线）表示在各判断条件下不同情况的决策路径。
- 有了该决策树后，比如买到一个西瓜，其特点是纹理是清晰，根蒂是硬挺的瓜，就可以判断是好瓜还是坏瓜：先看文理，文理清晰，往左走再看根蒂，根蒂硬挺，得到结论是坏瓜。

一. 选择特征

- 问题的关键是，怎么从样本数据出发构建出这个决策树结构。比如有样本数据：

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.46	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.36	0.37	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

- 根结点和每一层的内部结点，都是选择不同的特征，根据其特征的不同属性值进行划分，依次一直到叶子结点：



- 那么，在第一次选择特征的时候，为什么选择的是纹理，而不选择触感或其他呢？这也是构建决策树的关键步骤：以什么标准来选择特征。

1. 熵

- 特征选择是希望选取对训练数据具有更好分类能力的特征（分支后组内的数据更为 “一致”，或者叫 “纯度” 更高)，这样可以提高决策树模型的性能。下面就以西瓜分类为例阐释若干概念。
- “一致” 的对立面就是 “混乱”，“熵” 就是度量混乱不确定性的一个量，它越大越混乱，越小越 “一致”。
- 离散型随机变量 X 的概率分布为 $P(X=x_i)=p_i,\ i=1,\cdots,n,$ 则其**熵**定义为

$$H(X)=-\sum_{i=1}^{n} p_i \ln p_i$$

注：计算熵时以 2 为底还是以 e 为底只相差一个常数，不影响最小化，故机器学习领域都用 $\ln$.

- 对于数据集 D, $p_k = \frac{|D_k|}{|D|}$ 表示第 k 类样本所占的比例，则 D 的**经验熵**为：

$$H(D) = -\sum_{k=1}^{K} \frac{|D_k|}{|D|} \ln \frac{|D_k|}{|D|}$$

```r
library(tidyverse)
df = readxl::read_xlsx("datas/watermelon.xlsx")

calEntropy = function(Y) {  # 计算因变量 Y 分组下的经验熵
  p = table(Y) / length(Y)
  - sum(p * log2(p))
}

HD = calEntropy(df$好瓜)     # 计算好瓜的经验熵
HD
#> [1] 0.998
```

2. 条件熵

- 条件熵 $H(Y|X)$ 表示在随机变量 X 已知条件下随机变量 Y 的不确定性，定义为 X 给定条件下 Y 的条件概率分布的熵对 X 的数学期望：

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

其中，$p_i = P(X = x_i),\ i = 1, \cdots, n.$

- 计算特征 A 对数据集 D 的经验条件熵：

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i)$$

```r
calCondEntropy = function(A, Y) {
  # 计算特征 A 条件下结果变量 Y 的经验条件熵 H(Y|A)
  p = table(A) / length(A)
  H = tapply(Y, A, calEntropy)
  sum(p * H)
}

# 计算各个特征对结果变量 Y 的经验条件熵
HDA = map_dbl(df[2:7], calCondEntropy, Y = df$好瓜)
HDA
#>  色泽  根蒂  敲声  纹理  脐部  触感
#> 0.889 0.855 0.857 0.617 0.708 0.991
```

3. 信息增益

- **信息增益**表示因得知特征 X 信息而使得 Y 信息不确定性减少的程度。
- 特征 A 对数据集 D 的信息增益 $g(D, A)$，定义为数据集 D 的经验熵与特征 A 给定条件下 D 的经验条件熵之差（也叫互信息）：

$$g(D, A) = H(D) - H(D|A)$$

- 信息增益大的特征具有更强的分类能力。根据信息增益准则的特征选择方法是：对训练数据集（或子集）计算其每个特征的信息增益，选择信息增益最大的特征。

- 计算 6 种特征" 色泽", "根蒂", "敲声", "纹理", "脐部", "触感" 的信息增益:

```
gDA = HD - HDA
gDA
#>      色泽     根蒂     敲声     纹理     脐部     触感
#> 0.10813 0.14267 0.14078 0.38059 0.28916 0.00605
```

- 可见, "纹理" 的信息增益最大, 所以可选作第一个分类特征 (但仍不够科学)。

4. 信息增益率

- 作为绝对量，信息增益是有缺陷的，因为它对可取值较多的特征有所偏好！将信息增益除以该特征的经验熵做标准化，得到相对量就是信息增益率：

$$g_R(D,A)=\frac{g(D,A)}{H_A(D)}$$

其中，$H_A(D)=-\sum_{i=1}^{n_A}\frac{|D_i|}{|D|}\ln\frac{|D_i|}{|D|}$ 为特征 A 的经验熵, n_A 表示特征 A 的水平数。

```
HA = map_dbl(df[2:7], calEntropy)
gDA / HA
#>     色泽     根蒂     敲声     纹理     脐部     触感
#> 0.06844 0.10176 0.10563 0.26309 0.18673 0.00692
```

- 信息增益率结果表明，应该选择”纹理”作为决策树第一个分类特征。

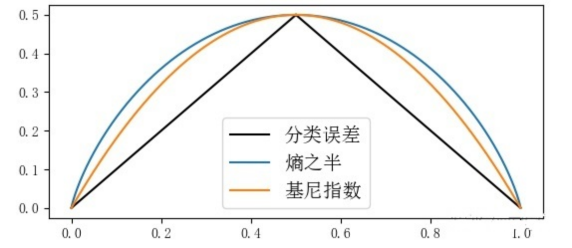
二. 决策树算法

- ID3 **算法**：根据信息增益选择特征，只能处理分类变量数据，没有剪枝的过程。
- C4.5 **算法**：根据信息增益率来选择特征，增加了剪枝过程，能够将连续特征离散化使用，能够处理缺失数据。
- **C5.0 算法**：是 C4.5 算法的改进版，引入了 Boosting、代价矩阵、代价敏感树等新技术，速度更快效果更好且更稳健，可以生成决策树或规则集（IF-THEN）。

注：这三种决策树算法都来自 Quinlan，都只能做分类，不能做回归。C50 包可以实现 C5.0 算法。

- 以上决策树算法，基于熵涉及大量对数运算，只能做分类。1984 年，Breiman 等提出了 CART（分类回归树）算法，用 Gini 指数来代替熵，递归地构建二叉树。
- 离散型随机变量 X 的概率分布为 $P(X = x_i) = p_i,\, k = 1, \cdots, n$, 则其 Gini **指数**定义为

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



- 可见，Gini 指数和熵之半的曲线非常接近，仅仅在 45 度角附近误差稍大。因此，Gini 指数可以做为熵的一个近似替代。

- 对于数据集 D，$p_k = \frac{|D_k|}{|D|}$ 表示第 k 类样本所占比例，则 D 的 Gini **指数**为：

$$Gini(D) = 1 - \sum_{k=1}^{K} \Big(\frac{|D_k|}{|D|}\Big)^2$$

- 若数据集 D 根据特征 A 是否取某一可能值 a 被分割成 D_1 和 D_2 两部分，则在特征 A 条件下，数据集 D 的条件 Gini 指数定义为：

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- $Gini(D)$ 表示数据集 D 的不确定性，$Gini(D, A)$ 表示经 $A = a$ 分割后数据集 D 的不确定性。Gini 指数越大，数据集的不确定性越大，这与熵类似。

```r
# 计算 Y 分组下的 Gini 指数
calGini = function (Y) {
  p = table (Y) / length (Y)
  1 - sum (p ^ 2)
}
```

```r
# 计算在特征 var 下，因变量 Y 分组下的条件 Gini 指数
CalCondGini = function(data, var, Y) {
  val = unique(df[[var]])
  n = length(val)
  rlt = vector("numeric", n)
  for (i in 1:n) {
    ind = df[[var]] == val[i]
    p = table(ind) / length(ind)
    g = tapply(df[[Y]], ind, calGini)
    rlt[[i]] = sum(p * g)
  }
  names(rlt) = val
  rlt
}
```

```r
map(names(df)[2:7], CalCondGini, data = df, Y = " 好瓜") %>%
  set_names(names(df)[2:7])
#> $ 色泽
#>   青绿   乌黑   浅白
#> 0.497 0.456 0.437
#>
#> $ 根蒂
#>   蜷缩   稍蜷   硬挺
#> 0.456 0.496 0.439
#>
#> $ 敲声
#>   浊响   沉闷   清脆
#> 0.450 0.494 0.439
#>
#> $ 纹理
#>   清晰   稍糊   模糊
#> 0.286 0.437 0.403
```

- 可见，特征 “**纹理**”= **清晰**的 Gini 指数 $= 0.286$ 最小，应该选择” 纹理” 作为第一个分类特征，根据其是否为” 清晰” 做二叉分支。
- 再依次对这两个分支分别做上述计算，选择下一层的二叉分支，依次类推直到所得节点都为叶结点。

2. CART 回归

- CART 回归是根据最小化均方误差进行特征选择，生成二叉树，也称为最小二乘回归树。
- 一棵回归树对应着特征空间的一个划分，比如 $R_1, \cdots, R_M$，以及在划分单元上的输出值 c_m, 则回归树模型可表示为：

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

其中，$I(\cdot)$ 为指示函数。用均方误差

$$\sum_{x_i \in R_m} \left(y_i - f(x_i)\right)^2$$

最小化原则求解每个单元上的最优输出值，即每个单元上所有输入 x_i 对应的输出 y_i 的均值。

- 问题的关键是如何对特征空间进行划分。采用启发式方法，选择第 j 个变量 $x^{(j)}$ 和其某个值 s，作为切分变量和切分点，定义两个区域：

$$R_1(j,s)=\{x:\ x^{(j)}\leq s\},\quad R_2(j,s)=\{x:\ x^{(j)}>s\}$$

  对固定输入变量 j, 通过求解：

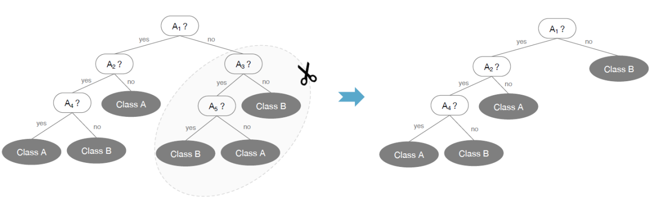
$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_i(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

  可得到最优切分点 s. 两个区域的输出值取区域所有输入 x_i 对应的输出 y_i 的均值。

- 遍历所有输入变量，找到最优的切分变量 j, 及相应的最优切分点 s，构成一个 $(j,\ s)$ 对。依次将特征空间划分为两个区域，对每个子区域重复上述划分过程，直到满足停止条件。

3. 决策树剪枝

- 决策树模型也有过拟合的问题（树过于复杂）使得模型泛化能力变差。
- 解决办法就是**剪枝**，即通过删除分支节点的子节点来更改模型，剪枝会减小树的大小，避免不必要的复杂性，从而解决过拟合问题。



- CART 用的是**后剪枝法**：
  - 从原始决策树生成各种剪枝效果的决策树；
  - 用交叉验证来检验剪枝后的预测能力，选择泛化预测能力最好的剪枝后的数作为最终的 CART 树。

- 对于带剪枝的决策树 T, 其损失函数为:

$$C_\alpha(T) = C(T) + \alpha|T|$$

  其中，$C(T)$ 表示模型对训练数据的预测误差，分类树用基尼指数，回归树用均方误差；$|T|$ 为决策树叶结点的数量，能够度量模型的复杂度，α 为正则化参数，同正则化回归，α 越大，则剪枝越厉害。

- 节点 T 剪枝前：$C_\alpha(T) = C(T) + \alpha|T|$; 节点 T 剪枝后：$C_\alpha(T') = C(T') + \alpha \cdot 1$. 令 $C_\alpha(T) = C_\alpha(T')$, 可得

$$\alpha = \frac{C(T') - C(T)}{|T| - 1}$$

- 计算出所有非叶节点的 α 值，分别对不同 α 所对应的剪枝后的最优子树做交叉验证，得到最佳的 α 及其对应的最优决策树。

- α 对应的超参数是，剪枝复杂度阈值（`cp`）。
- 其它提前停止条件：
    - 最小分支节点数（`minsplit`）
    - 最大深度（`maxdepth`）

4. 其它

- 决策树对缺失值不敏感，因为它可以将” 缺失” 当作一种特征值参与分类/回归，归入哪里对目标函数最优就归入哪里。
- 决策树模型的缺点：
  - 容易过拟合
  - 若数据不平衡，可能创建偏倚树
  - 模型不稳定性，数据的微小变化可能会导致生成完全不同的树
  - 决策树使用的贪婪方法不能保证得到全局最优决策树

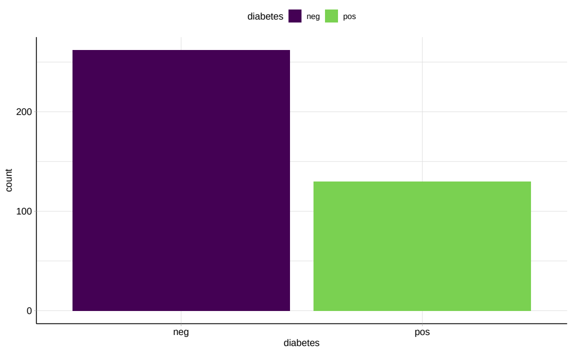
注：后两种缺点，可以通过集成学习方法解决（随机森林等）

- `mlr3verse` 实现决策树是调用 `rpart` 包。
- 使用来自 Kaggle 的印度糖尿病数据集 pima，包含 768 个样本、9 个变量，目标变量 diabetes 是二分类变量（是否患病），特征包括患者怀孕次数，BMI，胰岛素水平，年龄等。

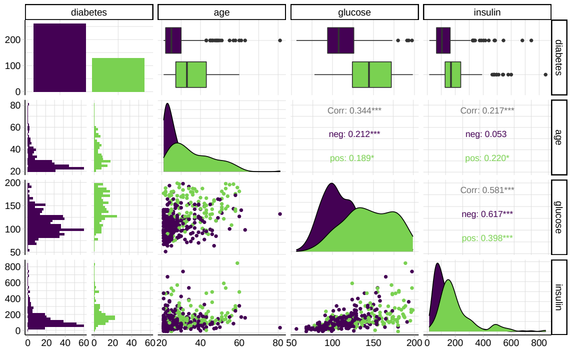
1. 创建任务

```
library(mlr3verse)
load("datas/pima.rda")
dat = na.omit(dat)
task = as_task_classif(dat, target = "diabetes")
task
#> <TaskClassif:dat> (392 x 9)
#> * Target: diabetes
#> * Properties: twoclass
#> * Features (8):
#>   - dbl (8): age, glucose, insulin, mass, pedigree, pregna
#>     triceps
```

```r
autoplot(task)
```



```r
autoplot(task$clone()$select(task$feature_names[1:3]),
         type = "pairs")      # 需要 GGally 包
```



2. 选择学习器

```
Rpart = lrn("classif.rpart" )     # 需要 rpart 包
Rpart
#> <LearnerClassifRpart:classif.rpart>: Classification Tree
#> * Model: -
#> * Parameters: xval=0
#> * Packages: mlr3, rpart
#> * Predict Types:  [response], prob
#> * Feature Types: logical, integer, numeric, factor, ordere
#> * Properties: importance, missings, multiclass, selected_f
#>   twoclass, weights
```

3. 划分训练集和测试集

- 做留出 (holdout) 重抽样，80% 作为训练集，其余 20% 作为测试集
- 为了保持训练集、测试集的因变量数据具有相似的分布，采用分层抽样方法
- 用 `partition()` 函数对任务做划分，默认按因变量分层，取出训练集索引和测试集索引

```
set.seed(123)
split = partition(task, ratio = 0.8)
# 默认 stratify = TRUE
```

4. 超参数调参

决策树模型有三个主要超参数：

- cp：剪枝复杂度阈值
- minsplit：最小分支节点数
- maxdepth：最大深度

```
Rpart$param_set                  # 查看超参数及默认值
#> <ParamSet>
#>                   id    class lower upper nlevels        def
#>  1:               cp ParamDbl     0     1     Inf
#>  2:       keep_model ParamLgl    NA    NA       2          F
#>  3:       maxcompete ParamInt     0   Inf     Inf
#>  4:         maxdepth ParamInt     1    30      30
#>  5:     maxsurrogate ParamInt     0   Inf     Inf
#>  6:        minbucket ParamInt     1   Inf     Inf <NoDefault
#>  7:         minsplit ParamInt     1   Inf     Inf
```

```r
library(paradox)
search_space = ps(
  cp = p_dbl(lower = 0, upper = 0.1),
  minsplit = p_int(lower = 1, upper = 20),
  maxdepth = p_int(lower = 1, upper = 30))

at = auto_tuner(
  learner = Rpart,
  resampling = rsmp("cv", folds = 10),
  measure = msr("classif.acc"),
  search_space = search_space,
  term_evals = 20,
  method = "random_search")
```

- 在训练集上启动调参过程

```
at$train(task, row_ids = split$train)
#> INFO  [13:03:36.700] [bbotk] Starting to optimize 3 parame
#> INFO  [13:03:36.743] [bbotk] Evaluating 1 configuration(s)
#> INFO  [13:03:36.770] [mlr3] Running benchmark with 10 resa
#> INFO  [13:03:36.810] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.837] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.860] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.882] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.904] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.927] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.949] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.970] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:36.994] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:37.021] [mlr3] Applying learner 'classif.rpar
#> INFO  [13:03:37.043] [mlr3] Finished benchmark
#> INFO  [13:03:37.084] [bbotk] Result of batch 1:
```

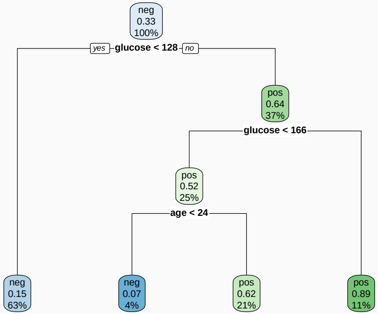
- 查看最优超参数

```
at$tuning_result
#>          cp minsplit maxdepth learner_param_vals  x_domain c
#> 1: 0.0433        2        7          <list[4]> <list[3]>
```

5. 模型训练

```
Rpart$param_set$values =
  at$tuning_result$learner_param_vals[[1]]
Rpart$train(task, row_ids = split$train)
```

```r
library(rpart.plot)
rpart.plot(Rpart$model, roundint = FALSE)
```



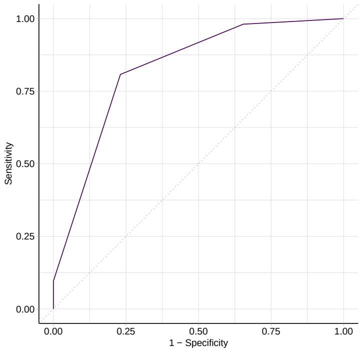
6. 模型预测

```
predictions = Rpart$predict(task, row_ids = split$test)
predictions
#> <PredictionClassif> for 78 observations:
#>     row_ids truth response
#>           8   neg      neg
#>          10   neg      neg
#>          25   neg      pos
#> ---
#>         369   pos      pos
#>         375   pos      pos
#>         376   pos      pos
```

```r
predictions$confusion                          # 混淆矩阵
#>          truth
#> response neg pos
#>      neg  42   6
#>      pos  10  20
predictions$score(msr("classif.acc"))   # 预测准确率
#> classif.acc
#>       0.795
```

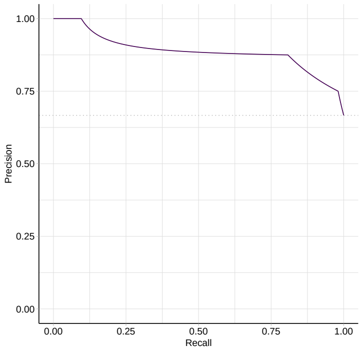
```r
Rpart$predict_type = "prob"
Rpart$train(task, row_ids = split$train)
predictions = Rpart$predict(task, row_ids = split$test)
predictions
#> <PredictionClassif> for 78 observations:
#>     row_ids truth response prob.neg prob.pos
#>           8   neg      neg    0.848    0.152
#>          10   neg      neg    0.848    0.152
#>          25   neg      pos    0.379    0.621
#> ---
#>         369   pos      pos    0.111    0.889
#>         375   pos      pos    0.379    0.621
#>         376   pos      pos    0.111    0.889
```

```r
autoplot(predictions, type = "roc")     # ROC 曲线
```



```r
predictions$score(msr("classif.auc"))   # AUC 面积
#> classif.auc
#>       0.825
```

```r
autoplot(predictions, type = "prc")     # PR 曲线
```



主要参考文献

[1] mlr3book. 2021. https://mlr3book.mlr-org.com/

[2] Jim Liang(梁劲). Getting Started with Machine Learning, 2019.

[3] 李航. 统计学习方法 (第二版). 清华大学出版社, 2019.

[4] 刘顺祥. 从零开始学 Python: 数据分析与挖掘. 清华大学出版社, 2018.