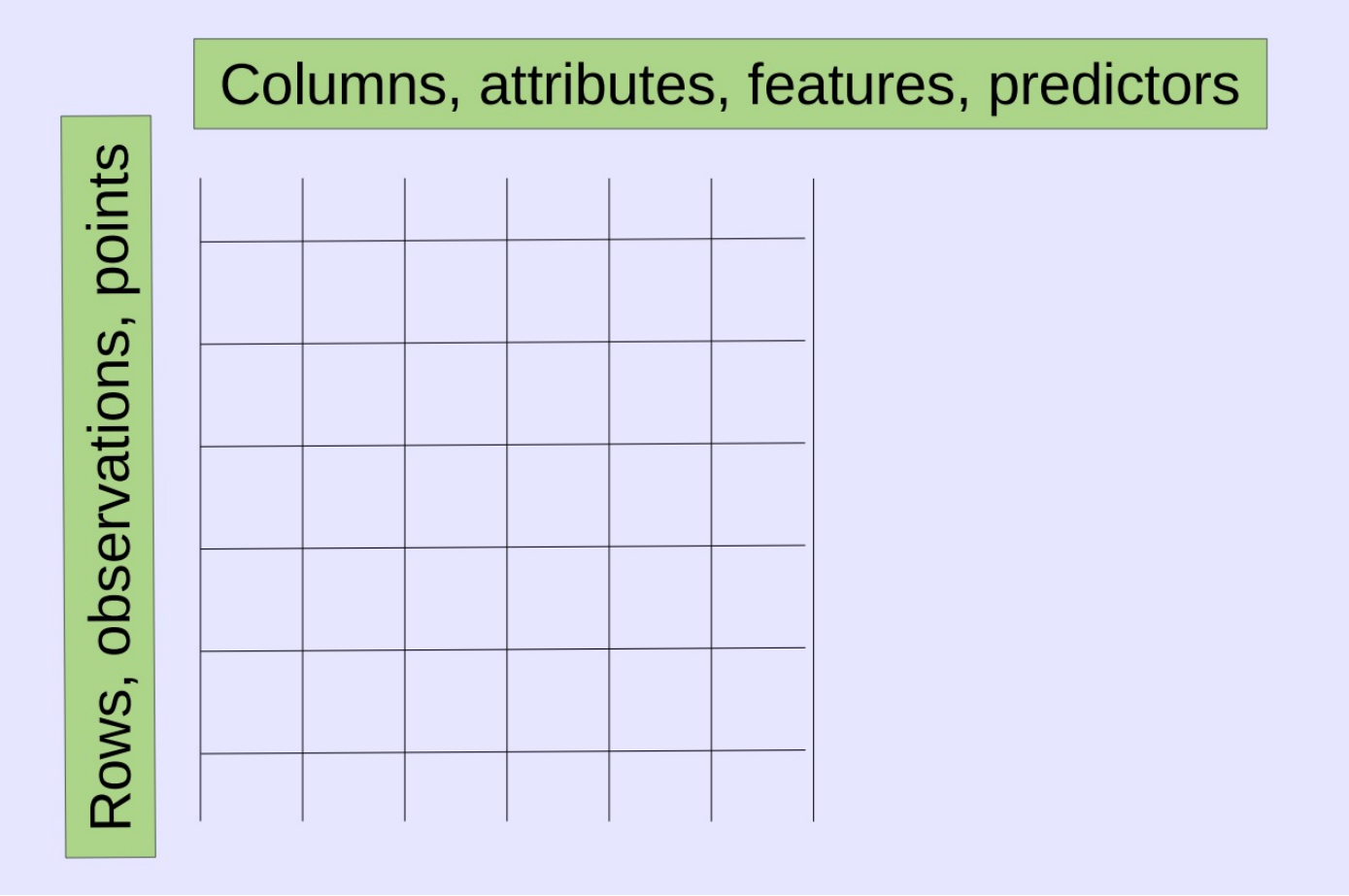
Components of Learning, Decision Trees 学习组成部分和决策树

- Topics:
  - Logistics and Recap
  - Follow instructions!
    - Decimal places
    - Read the questions
  - The dplyr package
  - Components of learning
  - Decision trees

组成部分

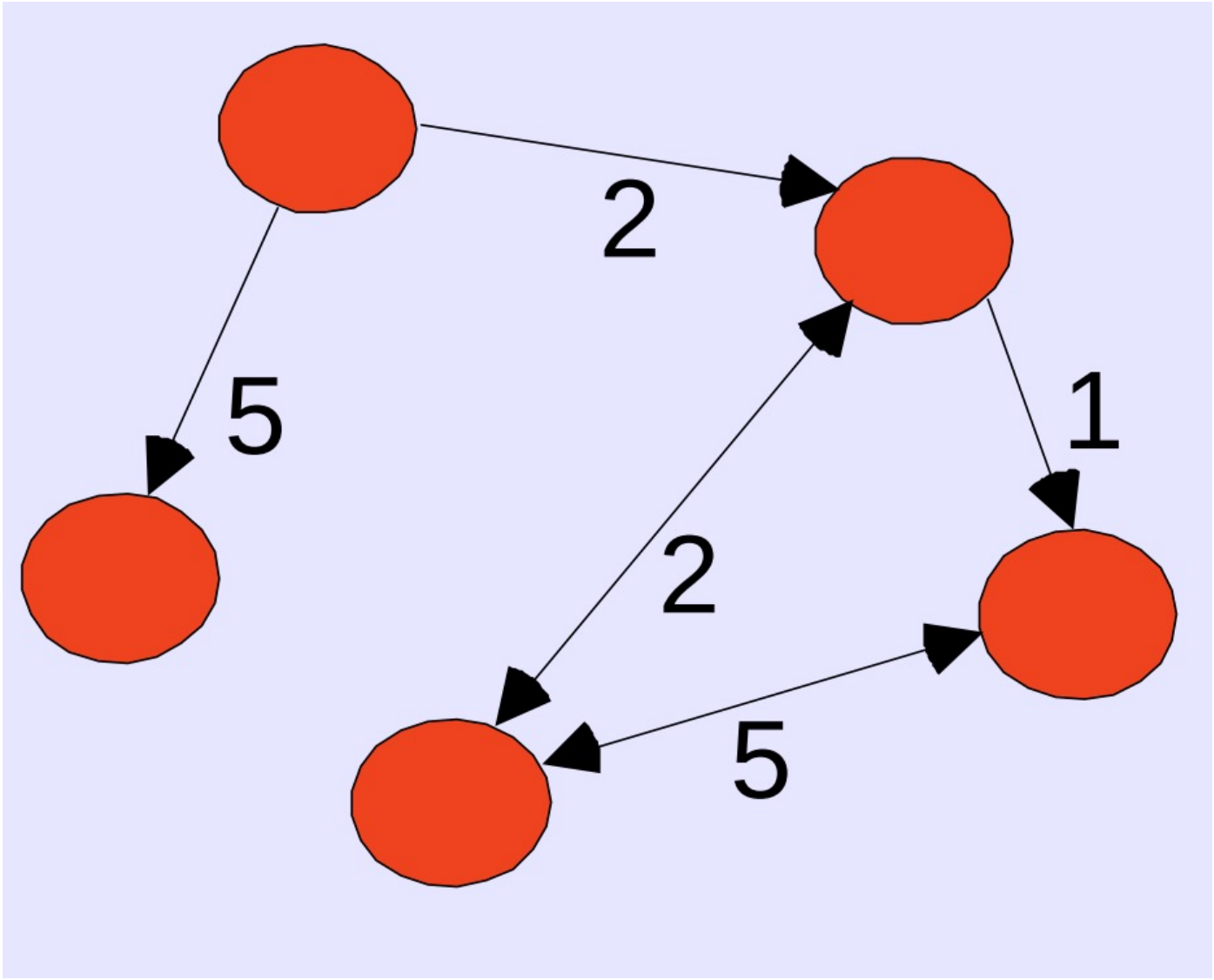
- 回想一下大多数的数据挖掘/机器学习算法是在矩阵中运行的
- 规范的样式是这样的:



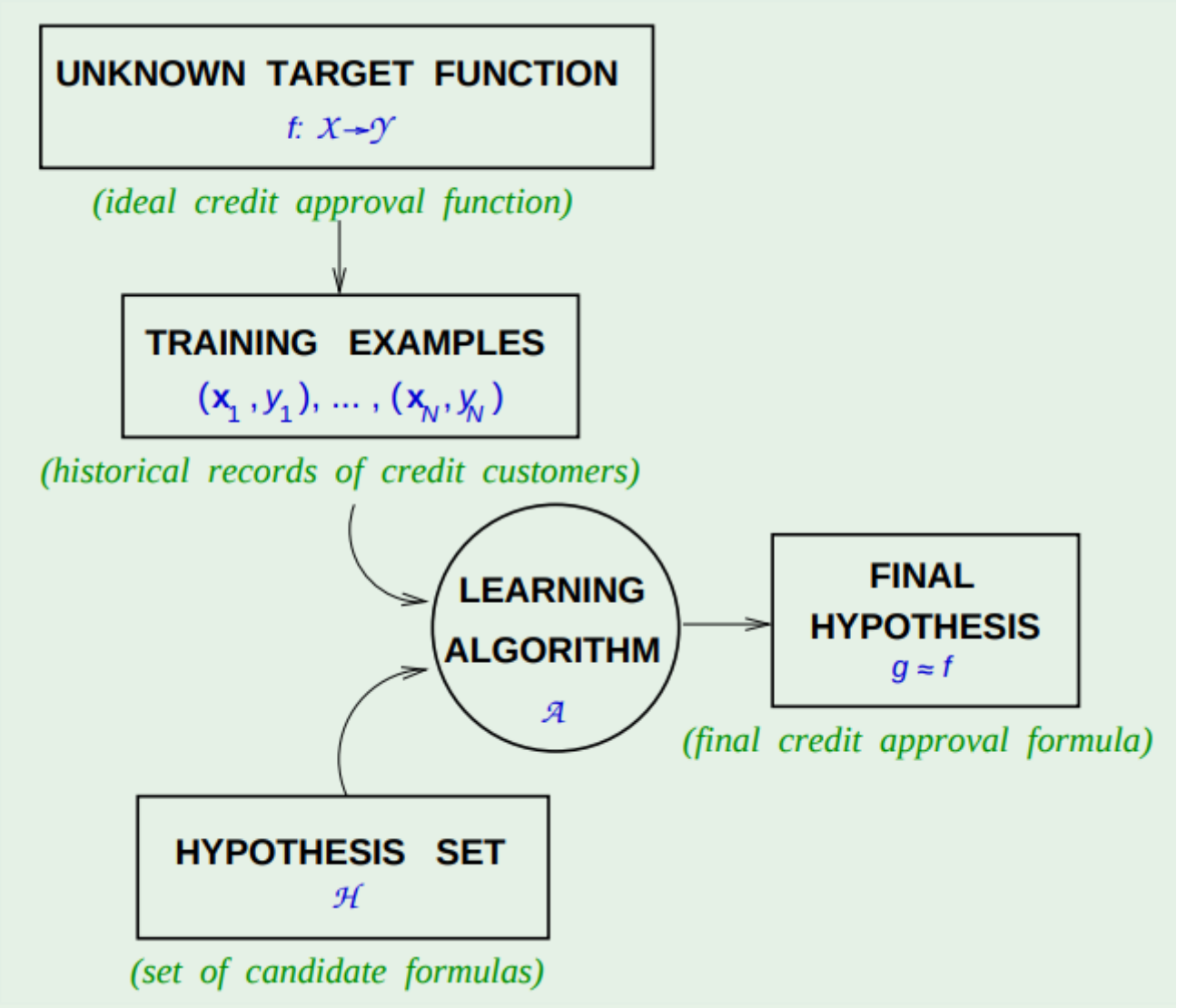
- 文档数据布局示例:

	token1	token2	token3	token4	token5	token6	token7	token8	token9	token10
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

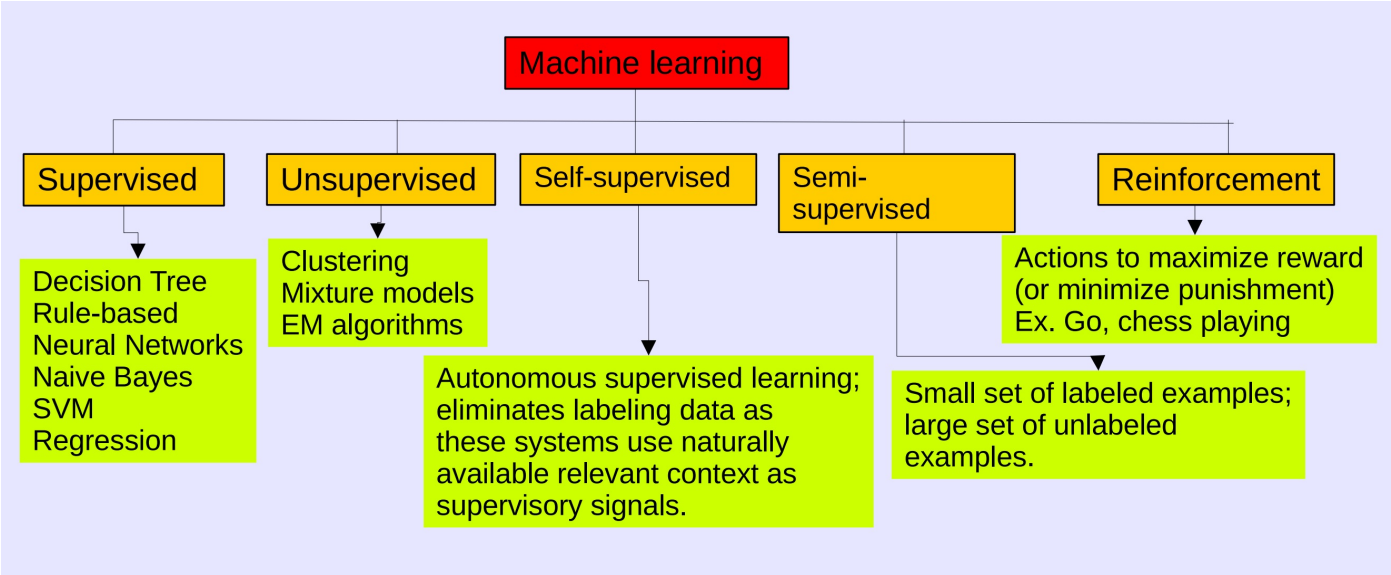
- 图数据布局示例:



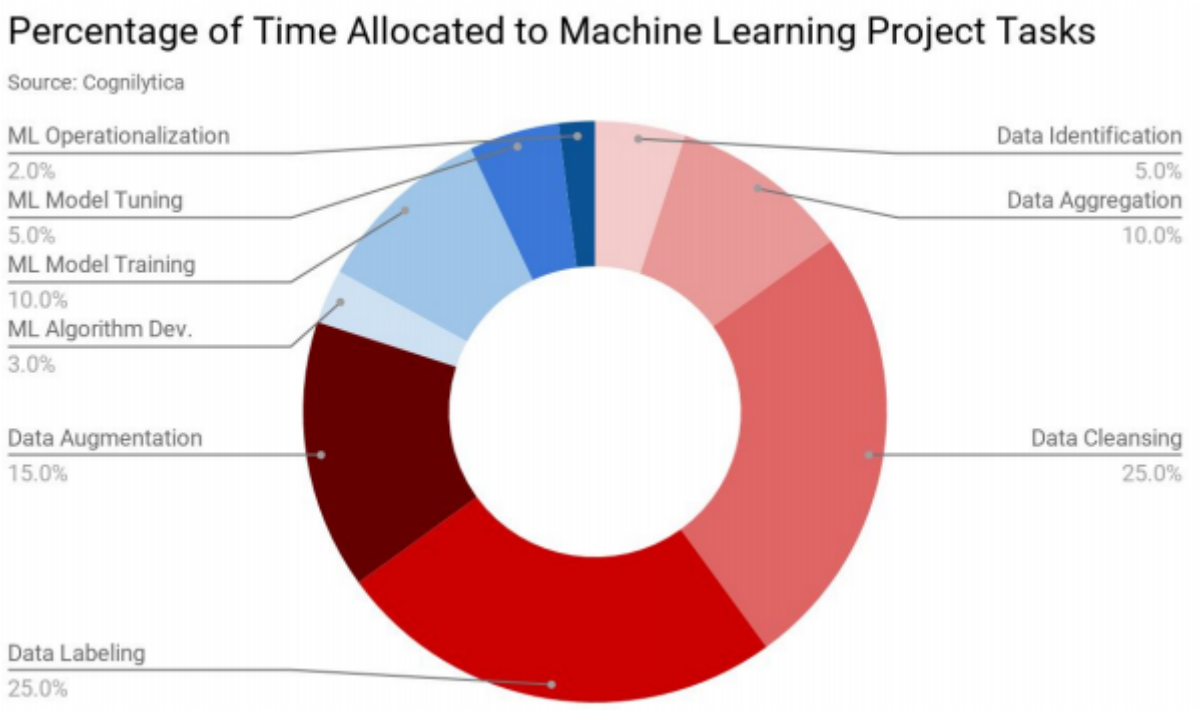
  - 结果证明，图可以展示为矩阵

- 公式表达
  - 输入:X,一个大于1个维度的矩阵
  - 输出:y,因变量
  - 目标函数:f:X->Y
  - 数据:(x_1, y_1),(x_2, y_2)...(x_n, y_n)
  - 假设:g:X->Y
  - 期望:g≈f
- 

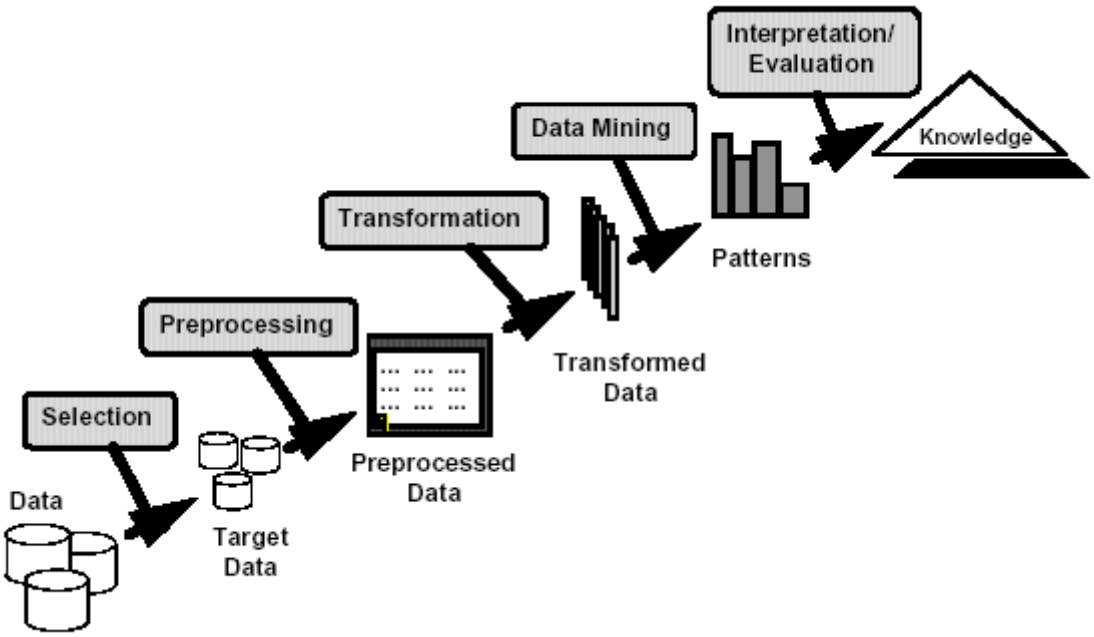
- 术语:学习器，分类器，模型
  - 学习器，通过x y的输入生成分类器
  - 分类器，通过上者的输入生成新y
  - 模型，由学习器生成，分类器通过模型来预测
-



- 数据科学家的时间分配



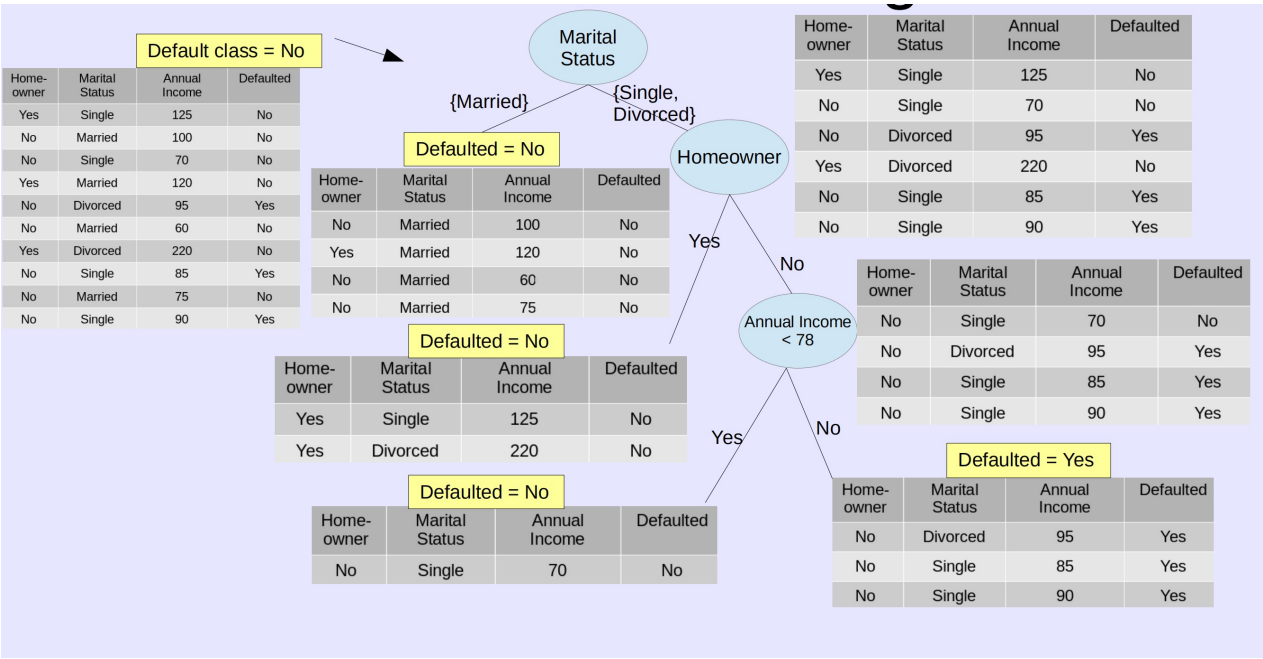
- 工作流



- R中有以下几种数据类型
  - Numeric 浮点数
  - Integer 整形
  - Factor 因子
    - 带序号因子 {“small”, “medium”, “large”}
    - 名词 {“blue”, “green”, “red”}
  - 字符串
- 特定算法需要特定的数据类型
- 在数据转化的时候需要确保后续的算法处理需要的数据类型可以对应上

决策树

- 理论部分
  - 我们的首个分类算法
  - 分类:目标是学习目标函数g，将每个属性集X映射到一个预定义的类标签Y
    - $f : x \to y \text{ where } x \in \mathbb{R}^n, \text{ and } y \in \mathbb{R}$
  - 例子
    -



- 默认分组;按照婚姻状态分组
  - 婚姻状态按照已婚/未婚+离婚
  - 是否有房屋
  - 收入分类

- Hunt's algorithm 亨特算法
  - Let D_t be the set of training records that reach a node t
  - General Procedure:
    - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
    - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
    - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

- 现在我们知道如何构建一个决策树，首先需要做到如何拆分数据
- 贪心策略
  - 基于属性测试优化结果来拆分数据

- 问题：
  - 如何拆分数据
    - 特定属性拆分
      - 取决于属性类型
        - 二元属性
        - 类别属性(不存在排序)
        - 类别属性(存在排序)
        - 连续值
          - 先排序
          - 将数据拆分n-1块，按区间分类

Defaulted	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420

Or IG (to be discussed).

      - 取决于分裂个数
        - 二元
        - 多元
    - 如何确定最优拆分
      - 熵:涉及的不确定性数量随机变量的值，或系统中的无序.
      - 熵值的公式定义:

$$H(Y) = -\sum_{i=0}^{c-1} P(Y = y_i)\ log_2 P(Y = y_i)$$

      - • Example:

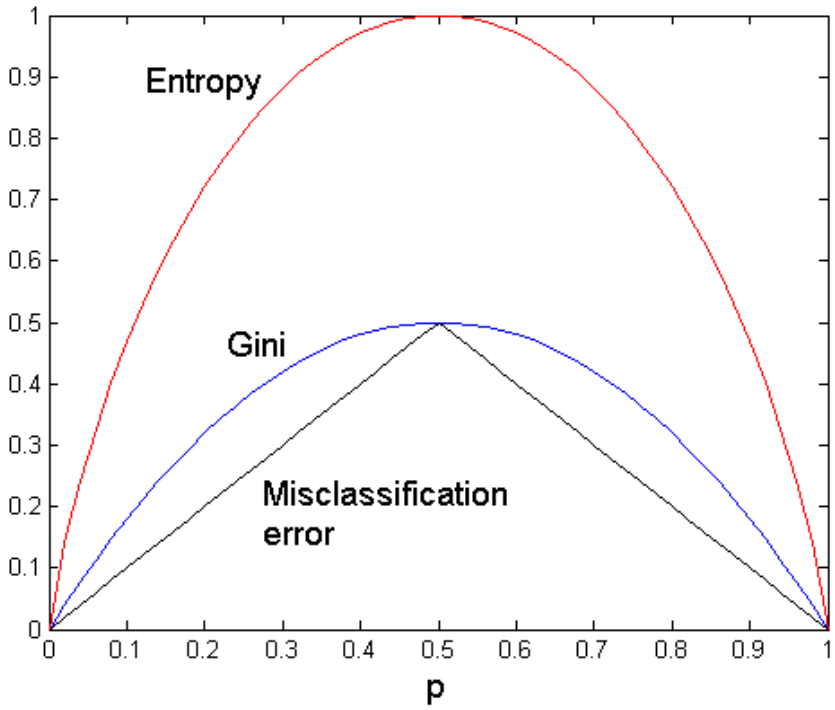
Tid	Home owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$H(Y) = -\sum_{i=0}^{c-1} P(Y = y_i)\ log_2 P(Y = y_i)$$

At the root node of the tree, Entropy is calculated as follows:

$$H(Y) = -[\frac{7}{10}\ log_2 \frac{7}{10} + \frac{3}{10}\ log_2 \frac{3}{10}] = 0.88$$

      - 在决策树算法中，熵值衡量纯度.
        - 纯度定义为节点内属于特定类的观察的分数
          - 如果观察值都属于同一类，我们有了纯净的结点，当我们有纯净的结点，我们就完成了最小化熵.
        - 衡量节点纯净度方法:

    - Gini index (使用CART和rpart包)
    - 熵(并不是概率问题)
    - 误分类错误
    - 

- 与熵关联的是信息 information
    - 信息的定义与熵完全相反 熵需要最小化，信息则需要最大化
- 所以当我们基于class拆分的时候，问题的关键在于这个class的属性中能给到多少information
    - 当时纯净分布时候，给我们最大的信息
    - 不相干的属性给我们很少信息
    - 所以在我们最大化信息的同时也在最小化熵
        - infomation gain(IG) = Entropy before the split - Entropy after the split
    - 在例子中我们需要基于Homeowner、Marital Status、Annual Income拆分，需要分别计算IG
        - 基于Homeowner拆分后的IG为:

Tid	Home owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$H(Y) = -\sum_{i=0}^{c-1} P(Y = y_i)\ log_2 P(Y = y_i)$$

At the root node of the tree, Entropy is calculated as follows:

$$H(Y) = -[\frac{7}{10}\ log_2 \frac{7}{10} + \frac{3}{10}\ log_2 \frac{3}{10}] = 0.88$$

Let's see what's the IG if we split on Homeowner attribute.

Tid	Home owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125K	No
4	Yes	Married	120K	No
7	Yes	Divorced	220K	No

$$H(Y) = -[\frac{0}{3}\ log_2 \frac{0}{3} + \frac{3}{3}\ log_2 \frac{3}{3}] = 0.0$$

Tid	Home owner	Marital Status	Annual Income	Defaulted?
2	No	Married	100K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$H(Y) = -[\frac{4}{7}\ log_2 \frac{4}{7} + \frac{3}{7}\ log_2 \frac{3}{7}] = 0.99$$

Now calculate the conditional entropy H(Y|X), or the remaining entropy of Y given X:

$$H(Defaulted|Homeowner) = \frac{3}{10} * 0 + \frac{7}{10} * 0.99 = 0.69$$

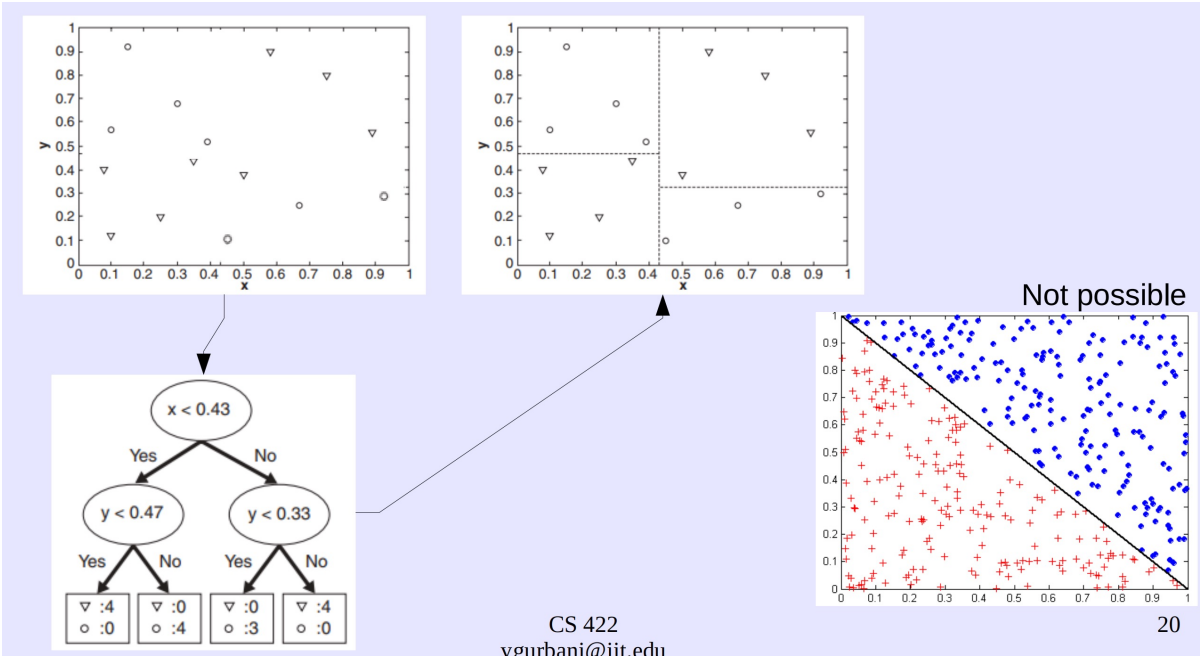
IG on splitting on Homeowner = Entropy before – Entropy after
= 0.88 – 0.69 = 0.19

- 基于Marital Status、Annual Income的拆分后IG为:

IG on splitting on Marital Status = Entropy before – Entropy after
= 0.88 – 0.60 = 0.28

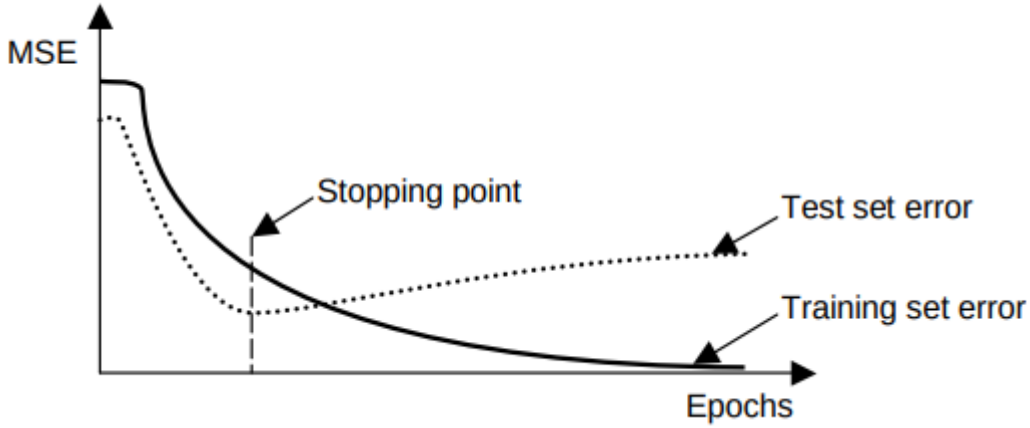
IG on splitting on Annual Income = Entropy before – Entropy after
= 0.88 – 0.69 = 0.19

- 何时结束拆分

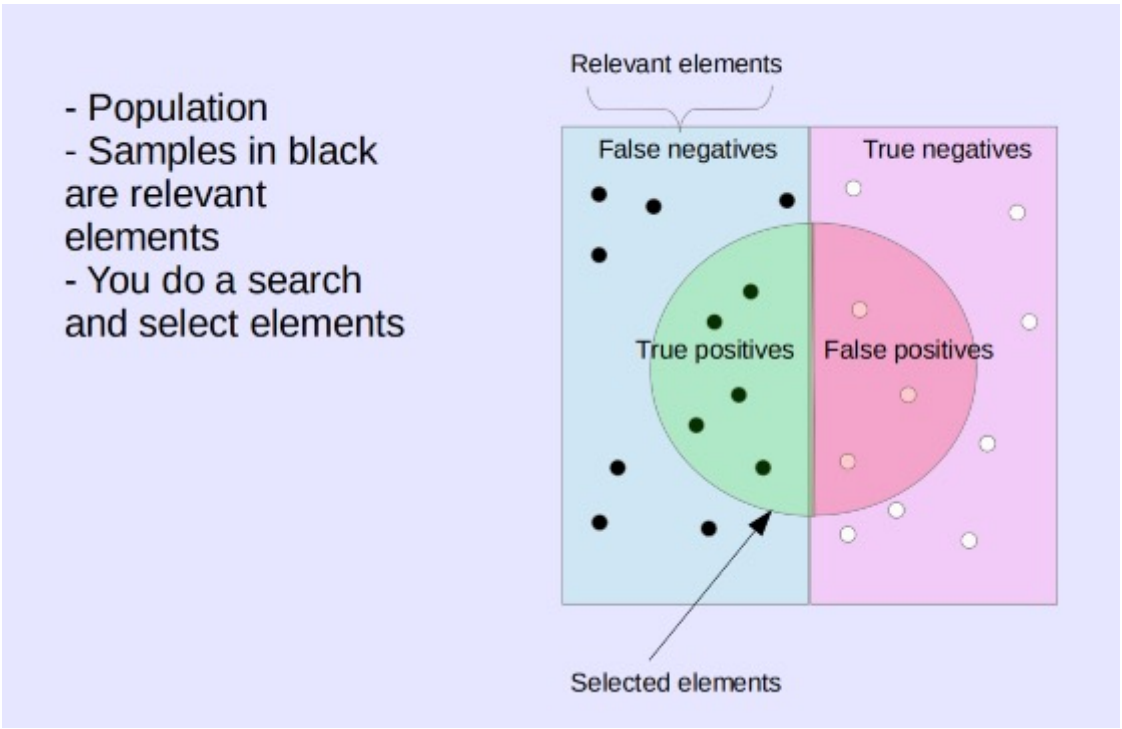
- 特征
  - 分类的非参数方法
  - 找到最优决策树NP
  - 构建最低成本树
  - 多重共线性(线性回归中的参数关联)不影响准确性
    - 正常情况下，如果发现属性间存在共线性，丢弃一个
  - 直线决策边界
  - 

- 模型选择

- 模型=算法+假设
- 当我们选择模型时，遵循以下步骤
  - 准备训练数据
  - 选择假设与算法
  - 调整算法
  - 训练模型，使用测试数据来验证结果
- 假设有两种模型(回归和神经网络)，那个模型更适合你的数据集
  - 评估模型适用性有两个指标
    - 模型检查 Model checking
    - 性能评估 Performance estimation
  - 最终目标，选择适用于数据集最好的模型
  - 我们如何定义最好？(最好的模型叫做最终模型)
  - 最好的模型是给你最小的预测错误或最小损失在训练集和测试集中
- 模型检查 给定一个数据集，我们将其分为训练集和测试集
  - 模型通过训练集训练，在测试集验证结果
  - 问题在于，如果随机出的测试集与模型并不具有代表性，相差过大
  - 解决方案:交叉验证 Cross validation
    - 交叉验证:一种系统地创建和评估多个模型的方法数据集的多个子集
    - 方法:
      - 留出法 Holdout method
        - 将数据集分为训练集、验证集、测试集
      - K折交叉验证 k-fold cross validation
        - 将数据集D划分成k个大小相似的互斥子集，每个子集Di都尽量保持数据分布的一致性，即从D中通过分层采样得到，然后每次用k - 1个子集的并集作为训练集，余下的那个自己作为测试集，这样可以获得k组训练/测试集，从而进行k次训练测试，最终结果为k次训练测试的均值
      - 留一交叉验证 Leave one out cross-validation (LOOC)
        - 留一交叉验证是一个极端的例子，如果数据集D的大小为N,那么用N-1条数据进行训练，用剩下的一条数据作为验证，用一条数据作为验证的坏处就是可能E_{val}和E_{out}相差很大，所以在留一交叉验证里，每次从D中取一组作为验证集，直到所有样本都作过验证集，共计算N次，最后对验证误差求平均，得到Eloocv(H,A)，这种方法称之为留一法交叉验证。
    - 在k-fold和LOOC数据验证中，将数据集拆分为3个部分更健壮
      - 训练集、验证集、测试集
    - 在这些留出方法中通过错误决定何时停止训练
      -



- 性能评估 在数据集或非正式的目标函数上训练的特定分类器
  - 如何评估模型的性能
    - 混淆矩阵 Confusion matrix 广泛运用
      - 提供矩阵计算的大量指标
    - ROC曲线 Receiver Operating Characteristics (ROC) curve
      - 表征正面命中和误报之间的权衡
  - 关注模型准确性，而不是训练时间，以及分类速度
  - 例子:



    - 对应的混淆矩阵

Confusion Matrix:

		Actual Class	
		Class = Yes	Class = No
Predicted Class	Class = Yes	TP	FP
	Class = No	FN	TN

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

From book; different format but same information as one on right.

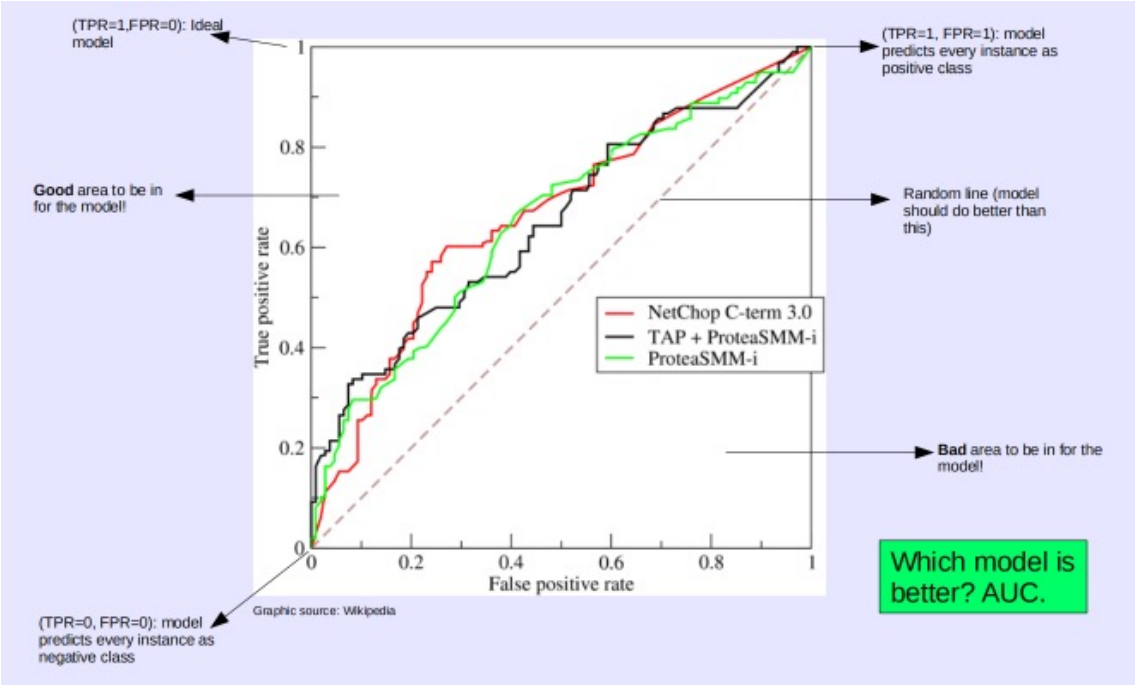
Classification accuracy: $\frac{(TP+TN)}{(TP+TN+FP+FN)}$

Error rate: $\frac{(FP+FN)}{(TP+TN+FP+FN)}$

- TPR true positive rate $TPR = TP/P = TP/(TP+FN)$
- TNR true negative rate $TNR = TN/N = TN/(TN+FP)$
- PPV positive predictive value $PPV = TP/(TP+FP)$
- 例子:在确诊为COVID-19的人群中，抗原检测在平均72%的有症状人群中正确识别出COVID-19感染。在没有感染COVID-19的人群中，抗原检测正确排除了99.5%有症状的人的感染
  - TP为72% FP为0.5% FN为28% TN为99.5%
  - TPR为0.72
  - TNR为0.995
  - 在这种情况下，这是个好的测试吗?
    - 就医护角度来说，这并不是，因为医护关注的是降低FN的值，也就是有28%比率有症状的人没有被识别出
- 独立的accuracy准确性并不是一个好的衡量标准
  -

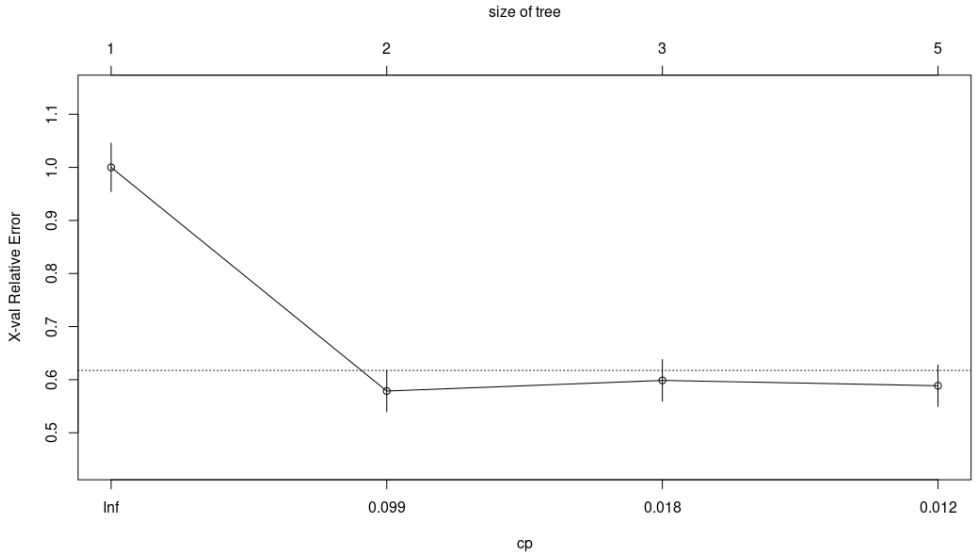
		Actual Class		Totals
		Class = Yes	Class = No	
Predicted Class	Class = Yes	TP 0	FP 0	0
	Class = No	FN 25	TN 125	150
Totals:		25	125	150

  - 在这里准确率很高，但是预测精确度为0
- 受试者工作曲线 Receiver Operation Characteristics(ROC) Curve
  -



- 选择最终模型
  - 最好的模型是提供最小预测错误(最少数据损失)在训练数据和测试数据
- 模型选择的最终确定
  - 在决策树中的过拟合和欠拟合
    - 如果我们让决策树深度增加，回增加过拟合风险
      - 可以通过修剪树来缓解过度拟合：将树生长到整个树，然后以自下而上的方式修剪节点.如果泛化错误得到改善，用叶子节点替换子树。 叶节点的类标签由子树中实例的多数类确定.
      - 修剪的两种方式
        - 预剪枝：基于某些约束（例如，杂质增益 < 阈值）停止树的生长.
          - 好处，树深度降低
          - 坏处，何时停止
        - 后修剪：将树增长到最大大小，然后修剪（例如，用新的叶节点替换子树，其类标签由与子树相关的大多数记录类确定。）
          - 好处，比预剪枝更好的结果，因为我们已经生成全部树
          - 坏处，可能浪费计算性能
      - 为了修剪，关注cp参数和与之关联的error和xerror，这两个参数与线性回归中的Multiple R-squared与Adjusted R-squared类似.
        - cp参数在rpart中定义为分割的阈值，因此不会尝试任何不会将整体失配度降低cp的分割
        - 任何不能通过cp改善拟合的拆分都可能会被交叉验证剪除，因此程序不需要继续它
        -

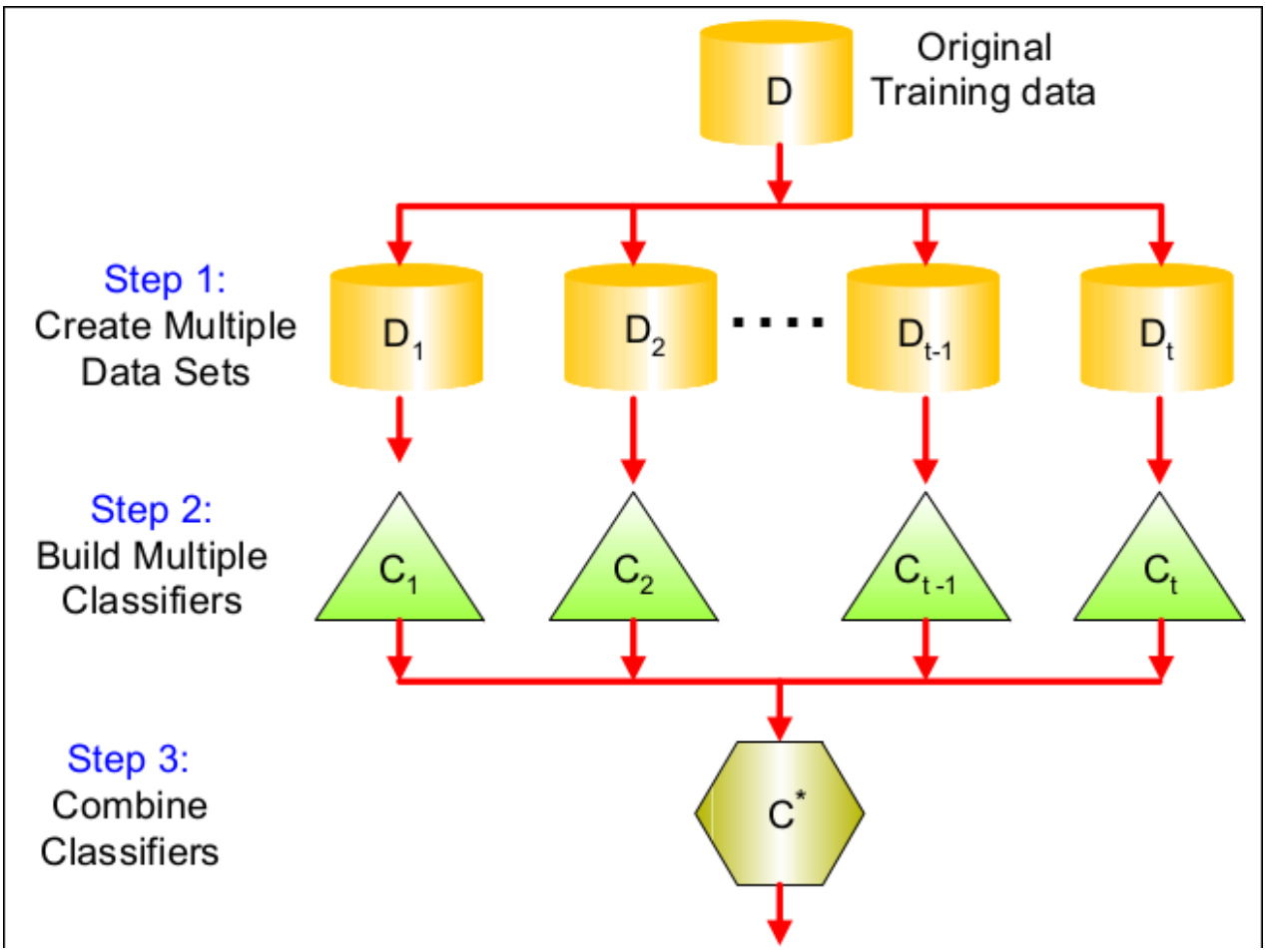
```
> printcp(model)

Classification tree:
rpart(formula = survived ~ pclass + sex + age, data = train,
    method = "class")

Variables actually used in tree construction:
[1] age    pclass sex

Root node error: 299/786 = 0.38

n= 786

      CP nsplit rel error xerror   xstd
1 0.4214      0     1.000  1.000 0.0455
2 0.0234      1     0.579  0.579 0.0388
3 0.0134      2     0.555  0.599 0.0393
4 0.0100      4     0.528  0.589 0.0391

> plotcp(model)
```




```
# Pruning the tree
printcp(model)
plotcp(model)
cpx <- model$cptable[which.min(model$cptable[,"xerror"]), "CP"]
pruned.model <- prune(model, cp=cpx)

# Run predictions on the pruned model
pred <- predict(pruned.model, test, type="class")
```

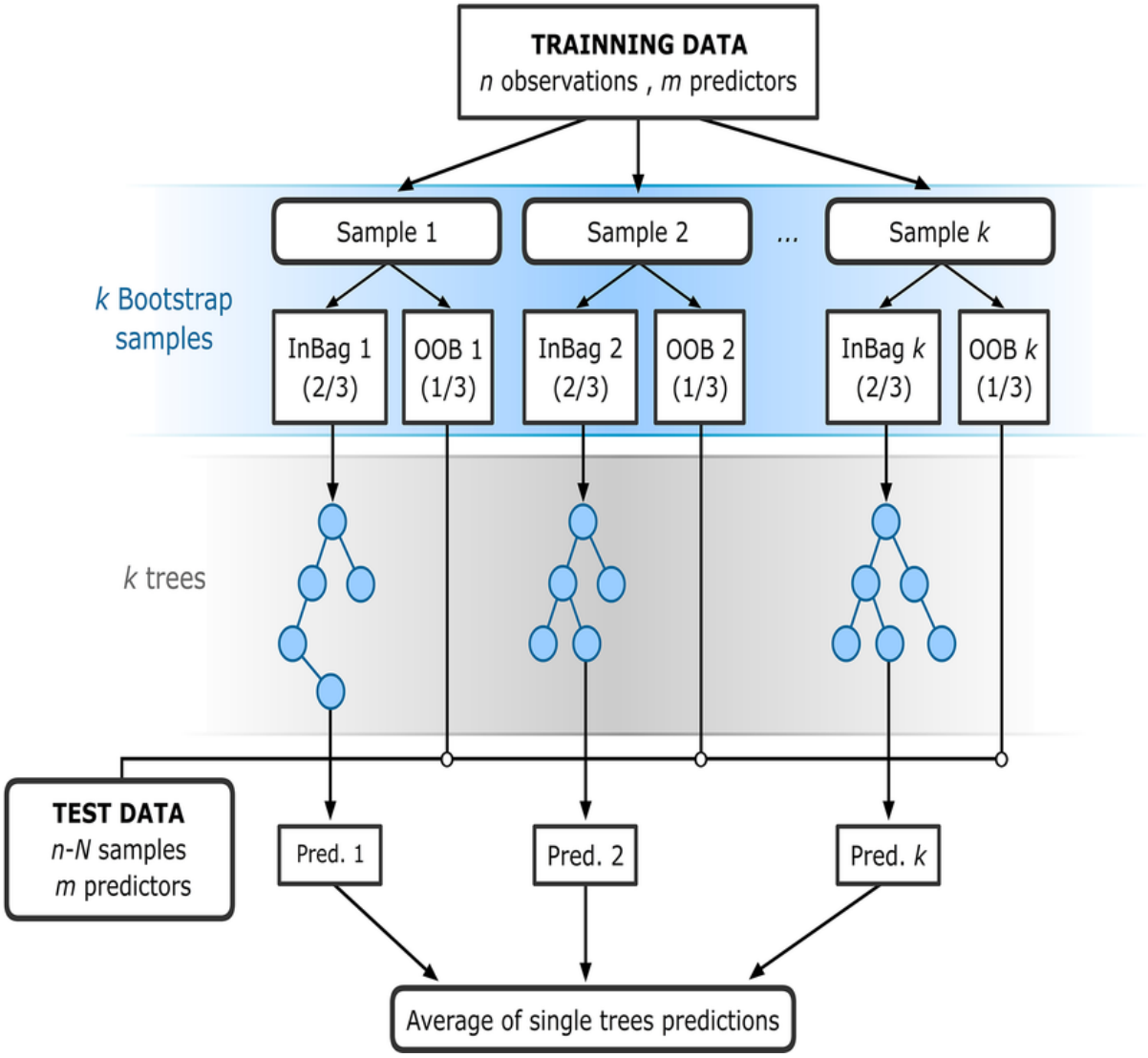
- 替代变量 Surrogate variables
  - ```
    Node number 1: 786 observations,    complexity param=0.4214047
      predicted class=0  expected loss=0.3804071  P(node) =1
        class counts:   487   299
       probabilities: 0.620 0.380
      left son=2 (518 obs) right son=3 (268 obs)
      Primary splits:
          sex    splits as  RL,           improve=102.305800, (0 missing)
          pclass < 1.5      to the right, improve= 30.798720, (0 missing)
          age    < 9.5      to the right, improve=  6.130452, (0 missing)
      Surrogate splits:
          age < 5.5      to the right, agree=0.662, adj=0.007, (0 split)
    ```
    ```

        - 变量重要性
            - ```
              rpart(formula = survived ~ pclass + sex + age, data = train,
                  method = "class")
                n= 786

                      CP nsplit rel error  xerror     xstd
              1 0.421405      0   1.00000 1.00000 0.045522
              2 0.023411      1   0.57860 0.57860 0.038848
              3 0.013378      2   0.55518 0.59866 0.039322
              4 0.010000      4   0.52843 0.58863 0.039088

              Variable importance
                 sex pclass    age
                  69     17     14
              ```
            - 所有的变量重要性总和为100，但是最重要的变量并不一定的是一个拆分元素
        - 如果我们停止过早，可能回欠拟合
- 集成方法 Ensemble methods
    - 到现在为止，我们从训练数据中包含了1种分类
    - 如果我们集成多种分类会带来更好的结果吗?
    - 

    - 通过训练数据建立多个分类器，在预测中使用结合函数
        - 投票模式
        - 逻辑回归
              ```

- 达到集成方法可以使用多种路径
  - 操纵训练数据
  - 操作输入端变量
  - 操纵类标签
  - 操纵算法学习
  - 使用多种不同学习算法
- 这些方法可以运行很好吗？(对于某些数据集)
  - 假设有25个分类器，每个的错误率低于0.35
  - 如果基分类相同的，每个都有相同的错误率0.35
  - 另一方面说，如果每个分类器独立，且错误不关联(这点很难做到，但是如果目标是为了降低错误率，这种方法可行)
  - 仅当>½的基分类器预测错误时，集成才会做出错误预测
  - P(X>=13)=

$$\sum_{i=13}^{25}\binom{25}{i}\varepsilon^{i}(1-\varepsilon)^{25-i}=0.06$$



- 类失衡 Class Imbalance
  - 对于大部分数据集来说类失衡是正常现象
    - 医疗数据
    - 制造业数据
  - 稀有（少数）类的正确分类比多数类的正确分类具有更大的价值
  - 类失衡给分类算法带来一系列问题
    - 准确性并并不能代表一个有效的衡量标准
    - 平衡准确度是更好的衡量标准，当测试或训练数据存在类不平衡
    - 缓解类失衡办法
      - 代价敏感学习，当模型犯了假阴性错误时，它会惩罚它
      - 抽样技术，确保失衡类分布正常
        - 减少主要类观察数据
          - 有用的数据可能不在样本中
        - 增加稀有类观察数据
          - 如果训练数据存在噪点，样本中也会存在噪点
        - 混合两种方法达到平衡数据集作用

        - 数据合成，如果可以，生成合成数据确保分布正常
- 多类决策树 Multi-class decision trees
    - 分类也扩展到区分多个类别
    - 通过整体准确率、每类精度和召回率评估多类回归模型
    - 在多类分组中建立混淆矩阵运用一对多原则
    - 涨势ROC曲线图，使用一对多原则将不同的类展示在图中