分类

黄建祺

目录

1 分类

分类是对一个实体进行分组的过程，类至少是一个相似的一组实体，这种相似性称为是特征，特征就是分类的基础。一个分类往往是从 0-1 分类问题开始，一开始我们有一系列的数据和属性，而我们希望通过属性的差异，找到一个方法能够将不同的类别区分开甚至实现预测某个特征属性下的类别。

1.1 分类的过程

分类在现实生活中无处不在，这种过程包括在人类知识结构中的每一个过程。分类是一种建立相互关系的过程，一种有目的的、系统的思考方式。从监督学习和无监督学习来看：分类属于一种有监督的学习策略。

1.2 数据挖掘与机器学习差异

机器学习是用于研究计算机的算法，将数据转换为智能行动。基于现有的数据、统计方法以及计算能力所发展而来的。而数据挖掘是从大型数据中找出有用的信息。机器学习更注重于数据来解决问题，而数据挖掘注重于计算机识别模式。

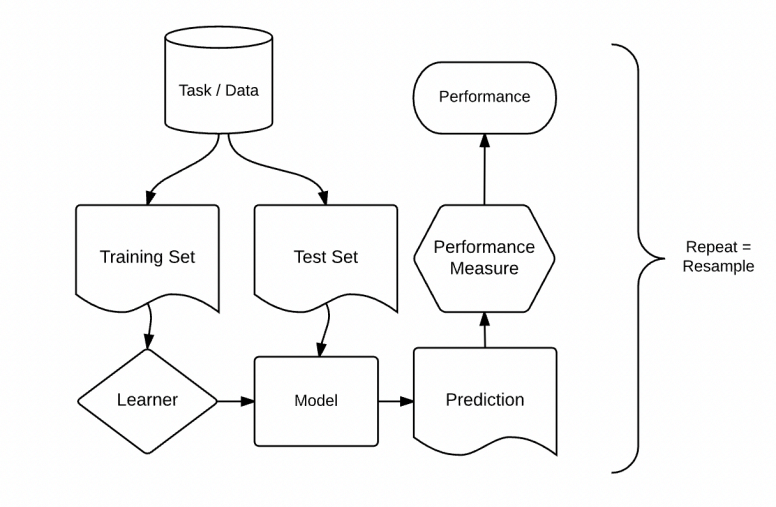


图 1: from mlr

1.2.1 距离度量

对于分类问题来说，距离是一个用于测度不同样本之间的关系以及对其进行划分的一个重要测量指标。最为常见的距离指标为欧氏距离

$$d(A,B)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2}$$

曼哈顿距离

$$d_1(A,B)=\sum_{i=1}^{n}|x_i-y_i|$$

余选相似度

$$cos(A,B)=\frac{<A,B>}{\|A\|\,\|B\|}$$

杰卡德相似系数

$$J(A,B)=\frac{A\cap B}{A\cup B}$$

即两个集合交与并之比。

1.3 训练与测试

进行训练与测试的原因，训练是为了通过统计学习得到一个模型，模型本身并不能仅仅服务于训练内的样本，而训练时候往往会使得模型服务于训练样本内的数据，因此测试是为了在防止训练的模型产生过拟合。

1.3.1 测试集、训练集、验证集

- 测试集: 用于训练模型内参数的训练集，模型直接根据训练集来对自身进行调整得到更好的预测效果。
- 验证集：用于训练过程之中检验模型的性能状态、收敛状况：
  - 常用于超参数调参、生成几组模型，根据模型在验证集上的表现来确定哪组超参数的更好的性能

  - 还可用于监督训练过程中的过拟合，以何时停止训练来实现。一般验证集稳定时后，继续训练，训练集的表现会得到提升，而验证集会不升反降，这时候就出现过拟合，可以终止。
- 测试集：用于评价模型的范化能力，之前确定的超参数用训练集调整之后，最后使用一个没有用过的来看能否胜任工作。

1.3.2 重抽样与交叉验证

常用的重抽样：- 留出法 (hold-out)：80 作为训练，20 作测试集 - k 折交叉验证 - 留一交叉验证 - Bootstrap 重抽样

1.3.2.1 嵌套重抽样 即两层的抽样：外层对整个数据重抽样，生成一组或多组的非测试集和测试集的划分，用于整体多次调参 + 拟合和评估性能。内层对非测试集重抽样，生成训练集合和验证集，用于监视训练过程。

而当我们划分不同的集合时候，不同的训练集合和测试集合的划分也会使得模型的准确率产生变化。因此为消除这一变化，可以创建一系列的训练集和测试集，计算模型在每一个测试集上的准确率之后取平均。也就是我们所说的 K 折交叉验证法。- 交叉验证的本质上是用不同的训练集/测试集对模型做多组的训练/测试，来应对单次的片面数据的问题。

K-cross validation 计算步骤：

1. 将原始数据集划分为相等的 K 部分（“折”）
2. 将第一部分作为测试集，其余作为训练集
3. 训练模型，计算模型在测试集上的准确率
4. 每次用不同的部分作为测试集，重复步骤 2 和 3 一共 K 次
5. 将平均准确率作为最终的模型准确率

1.3.3 CV 特例

留一法 LOOCV：属于交叉验证的特例，相当于将 k 等于 n 的交叉验证，每一个样本都被用于做测试，而剩余的数据 (n-1) 被用于做训练。

$$elppd_{loo}^{M} = \sum_{i=1}^{n} \log p(x_i|x_{-i})$$

其中的 loo 表示的是留一法，M_i 是某个模型，x_i 是某个测试数据。剩余的用于测试。很容易看出的是若一个模型的 elppd 越大，基于 (n-1) 的数据集的预测分布有越好的表现。

1.3.4 留一法的优缺点

- 并不受到划分方式的影响;
- 同时又存在计算量过大的问题

1.4 分类方法

1.4.1 回归

回归在任何的学科下都会得到相应的介绍。回归所涉及的是关于数值型因变量和一个或多个数值型自变量之间的关系。这里对一般线性回归与多元回归以及其参数不做多介绍。主要讨论关于回归树与模型树相关的问题

用于数值预测的决策树可划分为两类，一类称为决策树，20 世纪 80 年代的分类回归树 (classification and regression tree,CART)

另一类可称为模型树传统的回归对于数值预测是第一任务但是对于一些情况下，数值决策能够提供显著的优势。同样也涉及分割的标准：常见的是 SDR(standard deviation reduction)

$$SDR = sd(T) - \sum_i \frac{T_i}{T} \times sd(T_i)$$

$sd(T)$ 为集合 T 值的标准差，$T_1, T_2, \cdots, T_n$ 是对一个特征的一次分割的值的集合。$|T|$ 是集合 T 观测值的数量。这个公式实际上比较的是分割前后的标准差减少量。

1.4.1.1 非线性回归 其基本思路是先找到一个合适的参数模型能够将其转换为线性回归的形式。

$$N(t) = \frac{\varphi}{1 + e^{-\varphi_2 + \varphi_3 t}}$$

我们通过 `nls()` 来非线性拟合，寻找最优参数。非线性拟合依赖于对参数初始值的选取。选取适当的很快就能收敛到最优估计。否则迭代很可能无法收敛。

1.4.2 决策树

决策树是利用树形结构对特征和潜在结果之间建立联系的模型。基于这样的事实：能够对一个决策过程从上到下的实施，能够发现对最终预测的分类。#### 对决策树结构的刻画决策在根结点，遍历整个决策节点 (decision node)，决策节点要求的是基于工作属性做出选择。若可以做出选择，决策树在叶节点终止，叶节点所表示的是一系列决策而采取的行动。

1.4.2.1 分治法 决策树将数据划分为子集，不断分解直到达到某个准则下停止。

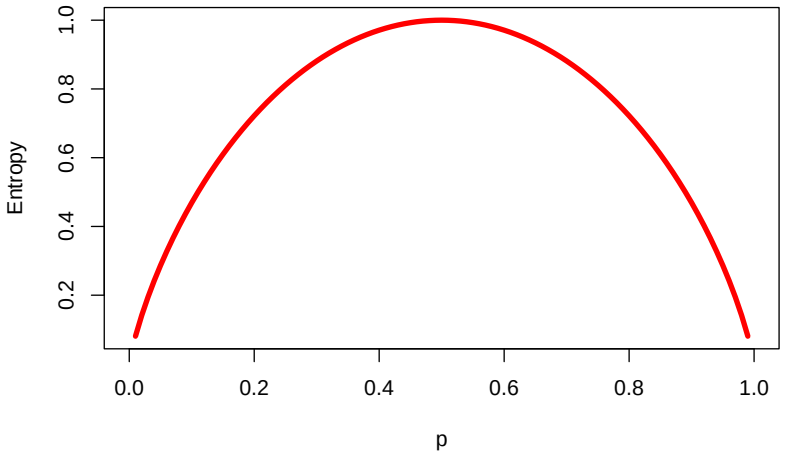
选择分割的过程：需要一个刻画分割的优劣程度的指标，我们这里仅以熵为例，熵的公式为:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

```
-0.6*log2(0.60)-0.40*log2(0.40)
```

```
## [1] 0.9709506
```

```
curve(-x*log2(x)-(1-x)*log2(1-x),col="red",xlab = "p",ylab = "Entropy",lwd=4
)
```



熵在 0.50 时候达到最大值，从一个局部到整体，往往会使用累加的形式，因此信息增益就是我们对整体熵值的刻画

$$InfoGain = Entropy(S_1) - Entropy(S_2)$$

而对于这决策树的信息增益的维度，仅仅在刻画整体划分的情况，而并没有对于决策树的复杂度进行刻画，因此若我们想要约束一棵树的复杂程度，我们可以使用所谓的“修剪决策树”的理念，具体通过提前停止法、预剪枝法等。

1.4.2.2 计算经验熵、信息增益和条件熵

条件熵就是在给定随机变量下的条件数学期望。

```
df = readxl::read_xlsx("r-ml-data/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)
HD
```

```
## [1] 0.9975025
```

```
library(purrr)
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)
```

```
}
HDA = map_dbl(df[2:7],calCondEntropy,Y= df$好瓜)
knitr::kable(HDA)
```

	x
色泽	0.8893774
根蒂	0.8548276
敲声	0.8567211
纹理	0.6169106
脐部	0.7083438
触感	0.9914561

```
gDA = HD - HDA
knitr::kable(gDA)
```

	x
色泽	0.1081252
根蒂	0.1426750
敲声	0.1407814
纹理	0.3805919
脐部	0.2891588
触感	0.0060465

1.4.3 LDA 方法

LDA is a supervised classification technique that is considered a part of crafting competitive machine learning models.

The original technique was developed in the year 1936 by Ronald A. Fisher and was named Linear Discriminant or Fisher's Discriminant Analysis. The original Linear Discriminant was described as a two-class technique. The multi-class version was later generalized by C.R Rao as Multiple Discriminant Analysis. They are all simply referred to as the Linear Discriminant Analysis.

The goal of LDA is to project the feature from a high dimension to a lower-dimensional space so that avoid the dimension catastrophe and also reduce resources and dimensional costs.

1.4.3.1 维度降低技术 (dimension reduction) 在多数情况下，特征空间经常是会存在冗余情况，因此我们可以降低原有的特征属性来实现帮助我们更好的预测，但实际上存在一个均衡，因为一般来说更多的特征有更好的预测效果，但我们要考虑到实际中对数据的处理能力。因此就需要进行维度缩减

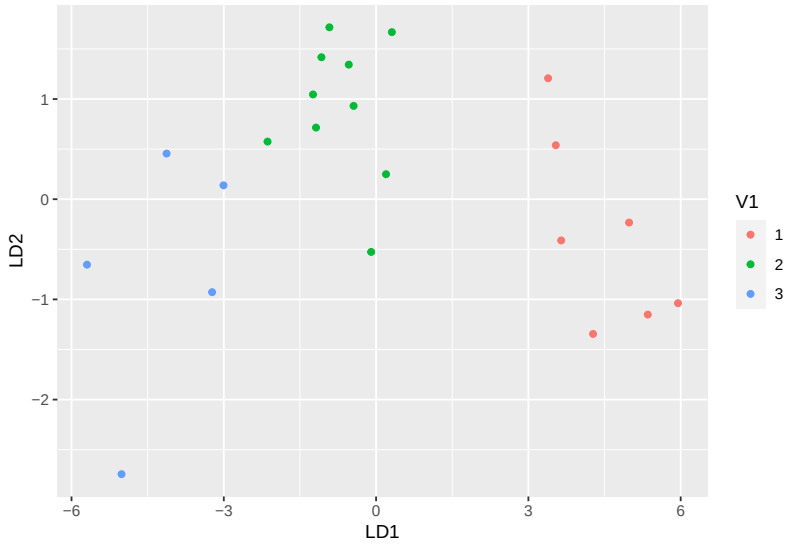
```
library(MASS)
rm(list=ls())
data<-read.csv("/Users/a182501/Desktop/大三上/data-mining/read_csv-lda.csv",header=FALSE,as.is = TRUE
```

我们同样可以使用贝叶斯理论在 LDA 方法中。

```
condata<-data[1:20,1:4]
grpdata<-data[1:20,5]
(con.sol=lda(condata,grpdata,prior = c(1,1,1)/3))
```

```
## Call:
## lda(condata, grpdata, prior = c(1, 1, 1)/3)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##        V1       V2        V3       V4
## 1 81.7000 16.56667 12.616667 47211.33
## 2 75.8375 13.96250  8.725000 17285.62
## 3 67.1500 11.70000  7.966667  7747.50
##
## Coefficients of linear discriminants:
##             LD1           LD2
## V1 2.401160e-01  1.597322e-01
## V2 2.175421e-01  3.708095e-01
## V3 2.397799e-01 -4.139835e-01
## V4 6.483192e-05 -5.180125e-05
##
## Proportion of trace:
##    LD1    LD2
## 0.9659 0.0341
```

```
ins<-data[21:22,1:4]
pred<-predict(con.sol,ins)
predall<-predict(con.sol,data[1:22,1:4])
df<-cbind(predall$class,predall$x)
df<-as.data.frame(df)
df$V1<-as.factor(df$V1)
ggplot(data = df)+
  geom_point(aes(x=LD1,y=LD2,color=V1))
```



最后我们会得到从 3 维到 2 维的转换。

1.4.4 朴素贝叶斯

1.4.4.1 理解概率 对于一对独立事件，是相互发生互不影响的过程。

$$P(A)P(B) = P(AB)$$

若所有事件是独立的，通过观测另一个事件就不能进行预测，因此相关事件 (dependent event) 就是用于刻画两个事件之间的相关性。但上述但公式并没有在一个事件发生的前提下考虑另一个事件的发生，仅仅在总体事件的概率维度上刻画，因此引入贝叶斯公式：

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

基于 A 事件的发生来刻画 B 事件发生的概率，也称为后验概率。朴素贝叶斯就是通过贝叶斯定理来刻画分类问题的方法。其基于一个基本假设：**在给定类别下，所有特征都具有相同的重要性和独立性**，各个输入特征 $X_i, i = 1, 2, \cdots, n$ 是相互独立的；

$$P(X_1, X_2, \cdots, X_n|C) = P(X_1|C)P(X_2|C)\cdots P(X_n|C) \tag{1}$$

利用贝叶斯公式我们可以计算出每个类别对应的概率 $P(C_K|X)$，其概率最大值为的 $P(C_k|X)$ 对应的类别，即为样本预测的类别

$$\hat{y} = argmax_{k\in\{1,2,\cdots\}} P(C_k|X)$$

其中分母为 $P(X) = \sum_{k=1}^{n} P(C_k)P(X|C_k)$ 对于所有的类别都是相同的。

而根据我们的贝叶斯独立性假设 (1) 可得到朴素贝叶斯

$$\hat{y} = \arg \max_{k\in\{1,\cdots,K\}} P(C_k) \prod_{i=1}^{n} P(X_i|k)$$

note: 特征间的独立性越强，分类效果越好。

1.4.4.2 对零频率的修正——拉普拉斯估计 对于一类事件的估计，我们在概率论计算条件概率时候往往使用的是链式法则，而这种法则的一大缺陷在于对某一种事件未发生过，进而影响其他事件对总体的判断。显然是较为不合理的。因此使用拉普拉斯估计，通过加入每类的每个特征中的一个发生次数 +1，能够保证上述问题得到解决。

```
library(mlr3verse)
library(readr)
dat = read_csv("r-ml-data/mushrooms.csv") %>%
  mutate(across(everything(), as.factor))
#glimpse(dat)
#summary(dat)
# 创建任务
task = as_task_classif(dat, target = "type")
task
```

```
## <TaskClassif:dat> (8124 x 22)
## * Target: type
## * Properties: twoclass
## * Features (21):
##   - fct (21): bruises, cap_color, cap_shape, cap_surface,
##     gill_attachment, gill_color, gill_size, gill_spacing, habitat,
##     odor, population, ring_number, ring_type, spore_print_color,
##     stalk_color_above_ring, stalk_color_below_ring, stalk_root,
##     stalk_shape, stalk_surface_above_ring, stalk_surface_below_ring,
##     veil_color
```

1.4.5 KNN

近邻分类器实际上是将无标记的案例分类为与它们最为相似的带有标记的所在的类。通过距离度量相似度：定位一个数据的近邻需要引入距离函数，能够用于度量两个实例之间的距离的函数。一般我们使用的是欧式距离 (Euclidean distance)。#### 选择 k k 是一个超参数，其选取需要考虑到训练数据时候的过拟合与欠拟合之间，也就是偏差-方差均衡 (bias-variance trade-off) 的关系。一个常见的方式是选择 k 为数据样本的平方根个数。但同样也需要进行多次尝试。其中我们就会发现，k 只是筛选位置距离，最终距离所有打标签样本仍然需要计算距离。若样本训练量较大 (假设为 N 个)，最终对于 M 个无训练样本需要进行 $N \times M$ 次运算。

1.4.5.1 懒惰学习 基于近邻的算法学习是懒惰的 (lazy learning) 算法。懒惰学习并非在学习，而是进行存储训练数据的过程。并没实际的训练，因此懒惰学习也称机械学习。

1.4.5.2 优缺点

- 简单高效

- 对数据分布无要求
- 训练阶段很快
- 不产生模型、理解特征与类如何相关的能力有限
- 需要一个合适的 k
- 分类阶段速度较慢

```
library(tidyverse)
library(mlr3verse)
dat = readxl::read_xlsx("r-ml-data/Knowledge.xlsx")
dat$UNS = as.factor(dat$UNS)# 将其转换为因子类型
knitr::kable(head(dat))
```

1.4.5.3 代码实现

STG	SCG	STR	LPR	PEG	UNS
0.00	0.00	0.00	0.00	0.00	Very Low
0.08	0.08	0.10	0.24	0.90	High
0.06	0.06	0.05	0.25	0.33	Low
0.10	0.10	0.15	0.65	0.30	Middle
0.08	0.08	0.08	0.98	0.24	Low
0.09	0.15	0.40	0.10	0.66	Middle

```
task = as_task_classif(dat, target = "UNS")
task
```

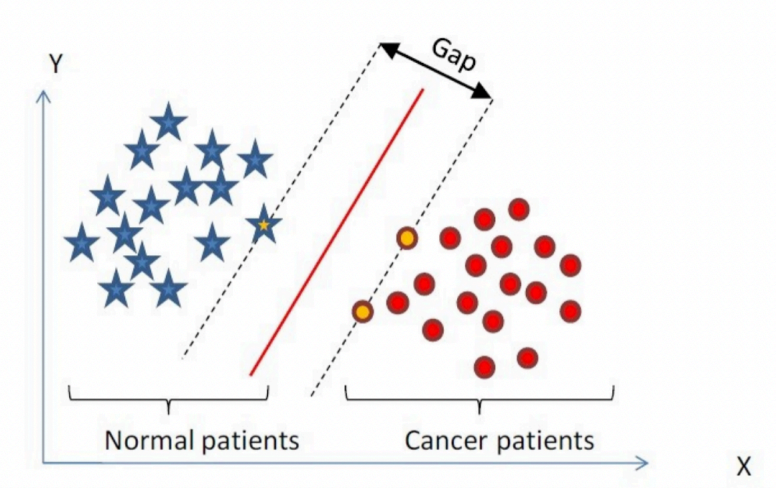
```
## <TaskClassif:dat> (403 x 6)
## * Target: UNS
## * Properties: multiclass
## * Features (5):
##   - dbl (5): LPR, PEG, SCG, STG, STR
```

```
knn = lrn("classif.kknn", predict_type = "prob",
  kernel = "rectangular", scale = FALSE)
knn
```

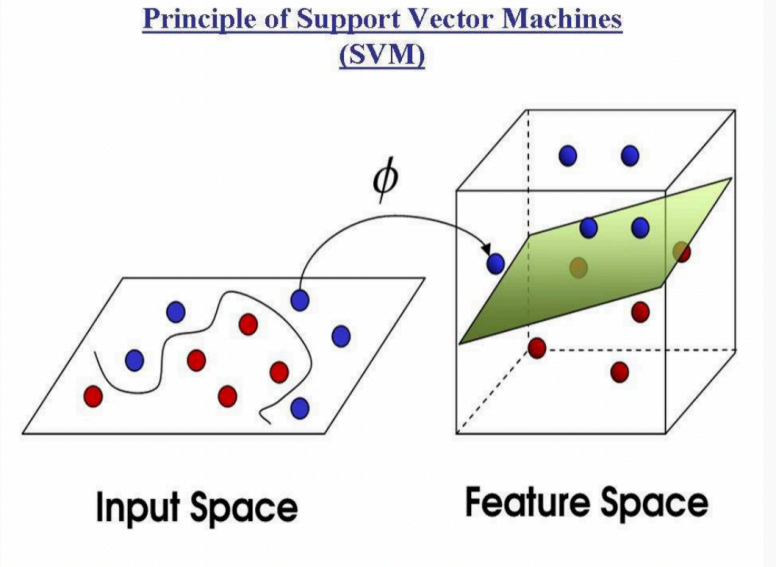
```
## <LearnerClassifKKNN:classif.kknn>
## * Model: -
## * Parameters: k=7, kernel=rectangular, scale=FALSE
## * Packages: mlr3, mlr3learners, kknn
## * Predict Types:  response, [prob]
## * Feature Types: logical, integer, numeric, factor, ordered
## * Properties: multiclass, twoclass
```

1.4.6 SVM 方法

经典的 SVM 方法是一种二分类模型，学习策略是希望间隔最大化，进而转为一个凸优化问题进行求解。



对于非线性的情况，SVM 的处理方法是选择一个核函数，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。



对原始特征内积的定义为 $< x, y >$，映射到 $\phi(x), \phi(y)$ 以后对于核函数来说

$$K(x, y) =< \phi(x), \phi(y) >$$

而核函数帮助我们跳过了中间对低维向高维的映射过程，直接可以进行转换得到高维映射空间。

1.5 模型评估

1.5.1 回归度量

- 对于回归问题，我们会使用 MSE 来对回归问题进行评估

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

进一步会将 MSE 开根号得到 RMSE

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

另一个度量：平均绝对误差 MAE

$$\frac{1}{n}\sum_{i=1}|y_i - \hat{y}_i|$$

以及对应的标准化下的**平均相对误差**

$$\frac{1}{n}\sum_{i=1}^{n}|\frac{y_i - \hat{y}_i}{y_i}|$$

可决系数 R^2

$$SSR = \sum_i(\hat{y} - \bar{y_i})^2 SSE = \sum_i(y_i - \hat{y}_i)^2 SST = SSR + SSE$$

R^2 所表示是对数据拟合优劣的评价：也就是能够在控制整体方差 (SST) 下的预测偏差最小。

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

分类度量评估一个分类模型的目的在于理解其对于未来的预测能力。- 对于分类问题，通常使用的是 error rate 进行评估

$$ErrorRate = \frac{1}{n}\sum_{i=1}^{n}(y_i \neq \hat{y}_i)$$

根据 MSE 和 errorrate 的表达式可以看出，实际上都是对于平均预测错误的可能性的度量。我们有时认为回归是属于分类的一种，但若将 ER 的方法用于回归判断的时候，$ErrorRate = \frac{1}{n}\sum_{i=1}^{n}(y_i \neq \hat{y}_i) = \frac{1}{n}\cdot n = 1$ 显然是较为不合适的，因此放宽非黑即白的假设将误差进行在绝对值意义上度量。

1.5.2 二分类的专用度量方式

1.5.2.1 准确率 (Accuracy) 所有的预测正确（正类负类）的占总的比重。

1.5.2.2 混淆矩阵 (Confusion Matrix) 对于一个二分类问题，预测结果与实际结果的两两组合会生成一个四种情况。也就是我们的混淆矩阵。

1.5.2.3 精确率 (Precision) 精确率是针对我们预测结果而言的，它表示的是预测为正的样本中有多少是真正的正样本。那么预测为正就有两种可能了，一种就是把正类预测为正类 (TP)，另一种就是把负类预测为正类 (FP)；

1.5.2.4 F 得分 (F-Score) F1 值为算数平均数除以几何平均数，且越大越好，将 Precision 和 Recall 的上述公式带入会发现，当 F1 值小时，True Positive 相对增加，而 false 相对减少，即 Precision 和 Recall 都相对增加，即 F1 对 Precision 和 Recall 都进行了加权。

1.5.2.5 ROC 曲线 在模型预测时候，这个预测值与阈值比较。将结果分为正类与负类。

```
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
data(aSAH)
head(aSAH)
```

```
##    gos6 outcome gender age wfns s100b  ndka
## 29    5    Good Female  42    1  0.13  3.01
## 30    5    Good Female  37    1  0.14  8.54
## 31    5    Good Female  42    1  0.10  8.09
## 32    5    Good Female  27    1  0.04 10.42
## 33    1    Poor Female  42    3  0.13 17.40
## 34    1    Poor   Male  48    2  0.10 12.75
```

```
R<-roc(aSAH$outcome, aSAH$s100b, smooth=TRUE,ci=T,auc = T)
```

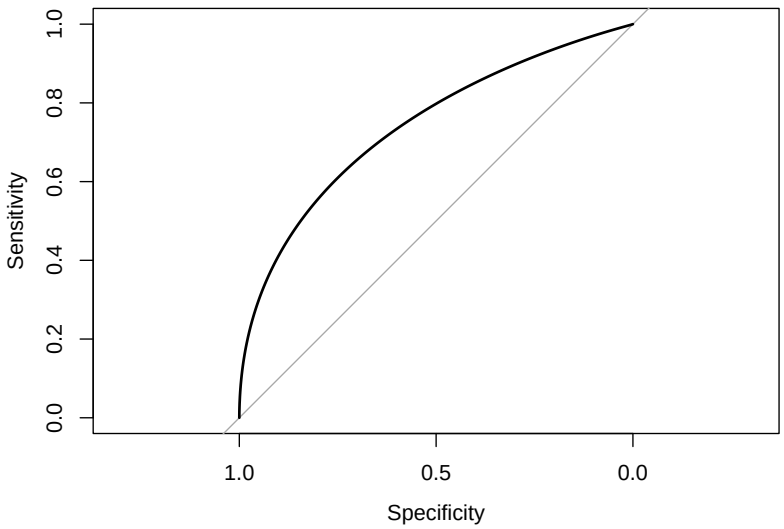
```
## Setting levels: control = Good, case = Poor
```

```
## Setting direction: controls < cases
```

```
R
```

```
##
## Call:
## roc.default(response = aSAH$outcome, predictor = aSAH$s100b,     smooth = TRUE, auc = T, ci = T)
##
## Data: aSAH$s100b in 72 controls (aSAH$outcome Good) < 41 cases (aSAH$outcome Poor).
## Smoothing: binormal
## Area under the curve: 0.74
## 95% CI: 0.633-0.8302 (2000 stratified bootstrap replicates)
```

```
plot(R)
```

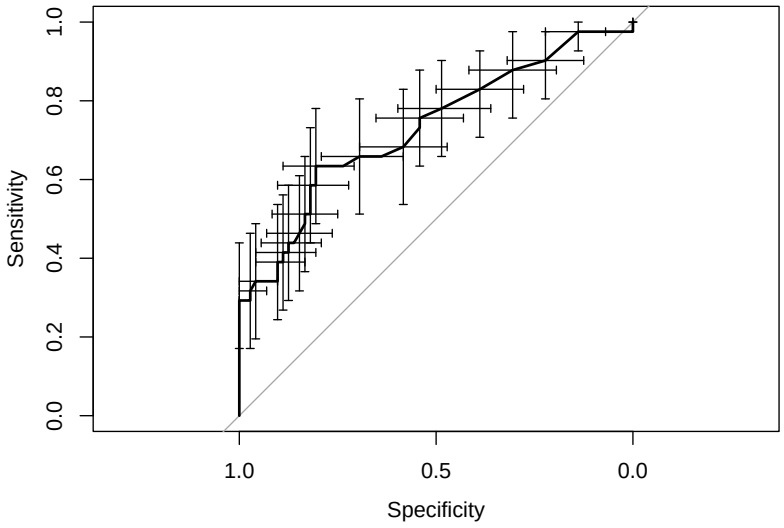


在 pROC 包中，可用 plot.roc 绘制置信区间

```
plot.roc(aSAH$outcome, aSAH$s100b,
         ci=TRUE, of="thresholds")
```

```
## Setting levels: control = Good, case = Poor
```

```
## Setting direction: controls < cases
```



1.5.2.6 AUC AUC 实际上为 ROC 曲线下面积值。量化地反映基于 ROC 曲线衡量出的模型性能。AUC 越大说明分类性能越好。