性能评价与度量

模型评估的意义在于提高模型的预测精度和泛化能力
为了评估机器学习算法在某项任务中好坏，需要我们设计方法去度量性能

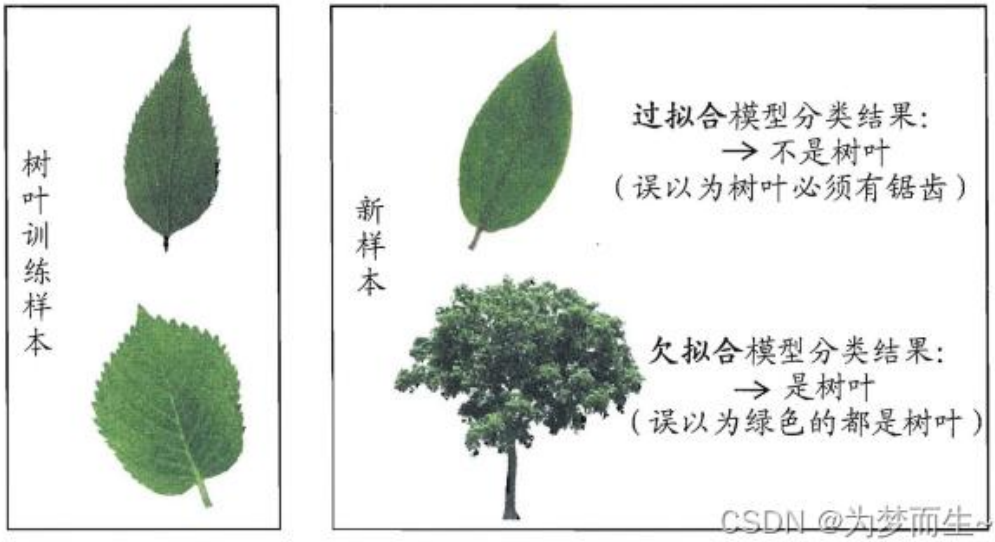
在模型建立中，通常有两个数据集：**训练集**（train）和**测试集**（test）。**训练集**用来训练模型；**测试集**是完全不参与训练的数据，仅仅用来观测测试效果的数据，验证模型在新数据上的泛化能力。

一般情况下，训练的结果对于训练集的拟合程度通常还是挺好的，但是在测试集总的表现却可能不行。比如下面的例子：

图一的模型是一条线型方程。 可以看到，所有的红点都不在蓝线上，所以导致了错误率很高，这是典型的不拟合的情况
图二 的蓝线则更加贴近实际的红点，虽然没有完全重合，但是可以看出模型表示的关系是正确的。
图三，所有点都在蓝线上，这时候模型计算出的错误率很低，（甚至将噪音都考虑进去了）。这个模型只在训练集中表现很好，在测试集中的表现就不行。 这是典型的‘过拟合’情况。

过拟合与欠拟合

当学习器过于适应训练样本时，它可能会将训练样本自身的某些特点视为所有潜在样本都具有的一般性质，从而导致泛化性能下降。这种现象在机器学习中称为“过拟合”。与之相对的是“欠拟合”，这意味着学习器尚未完全掌握训练样本的一般性质。



泛化性能（Generalization Performance）

人工智能(AI)的发展目标之一就是让机器具备泛化能力
泛化能力是指一个系统在未知情况下能够进行有效决策和推理的能力.
适应性强：具有泛化能力的 AI 系统可以在新的环境和任务中快速适应，而不需要大量的人工干预。
可扩展性好：具有泛化能力的 AI 系统可以在不同领域和任务中得到广泛应用，而不需要重新设计和开发。
可维护性好：具有泛化能力的 AI 系统可以在新的数据和任务中保持高效性能，而不需要不断地更新和优化。

主要的评估方法包括内部评估和外部评估。内部评估主要基于训练集和测试集进行，如准确率、查准率、查全率和 F1 分数等。
外部评估则通过比较模型与其他基准模型的性能来进行评估，如 ROC AUC 面积和交叉验证等。
此外，为了充分了解模型的性能，还常常采用交叉验证等技术进行评估。

混淆矩阵

		预测的类		
实际的类		类 = 1	类 = 0	
	类 = 1	TP	FN	P
	类 = 0	FP	TN	N

P(Positive Sample)：实际正类
N(Negative Sample)：实际负类

TP(True Positive)：正确预测到的正例的数量。
FP (False Positive)：把负例预测成正例的数量。
FN (False Negative)：把正例预测成负例的数量。
TN(True Negative)：正确预测到的负例的数量。

已知条件：班级总人数100人，其中男生80人，女生20人。

目标：找出所有的女生。

结果：从班级中选择了50人，其中20人是女生，还错误的把30名男生挑选出来了。

	相关(Relevant),正类	无关(NonRelevant),负类
被检索到 (Retrieved)	true positives(TP 正类判定为正类,例子中就是正确的判定"这位是女生")	false positives(FP 负类判定为正类,"存伪",例子中就是分明是男生却判断为女生,当下伪娘横行,这个错常有人犯)
未被检索到 (Not Retrieved)	false negatives(FN 正类判定为负类,"去真",例子中就是,分明是女生,这哥们却判断为男生--梁山伯同学犯的错就是这个)	true negatives(TN 负类判定为负类,也就是一个男生被判断为男生,像我这样的纯爷们一准儿就会在此处)

TP = 20; FP = 30; FN = 0; TN = 50;

准确率 （accuracy)

准确率：衡量正确预测的样本占总样本的比例。
计算方式为 ：

$$\mathbf{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

查准率（Precision）

查准率：也称精确率，表示正确分类的正例个数占被预测为正例的实例个数的比例
计算方式为：

$$\mathbf{Precision} = \frac{TP}{TP+FP}$$

查全率（Recall）

查全率：也称召回率，表示正确分类的正例个数占实际正例个数的比例
计算方式：

$$\mathbf{Recall} = \frac{TP}{TP+FN}$$

查准率 P $P=\frac{TP}{TP+FP}$　　查全率 $R=\frac{TP}{TP+FN}$

$P=\frac{\text{预测正确的正类}}{\text{所有预测出来的正类}}$　　$R=\frac{\text{预测正确的正类}}{\text{真实情况中所有的正类}}$

真实情况	预测结果	
	正例 西瓜(10)	反例 橘子(10)
正例 西瓜(10)	TP(真正例) 10(真的是西瓜)	FN(假反例) 10(不是橘子是西瓜)
反例 橘子(10)	FP(假正例) 20(不是西瓜,是橘子)	TN(真反例) 20(真是橘子)

模型的查全率和查准率的高低会是模型好坏评判的一部分

一般来说，查准率高时，查全率往往偏低
而查全率高时，查准率往往偏低

P高时 需严格筛选结果 FP↓可能遗漏部分正例导致 FN↑ ⇒R↓
R高时 需尽量包含正例 ⇒可能引入更多反例FP↑ ⇒P↓

对一个算法的评估 需要综合考虑查准率和查全率的性能度量 ⟶ 平衡点、F1度量

F1度量　(查全率和查准率做一个平均)

调和平均 ⟶ n个数的倒数的算数平均数

$$\frac{1}{F_1}=\frac{1}{2}\cdot\left(\frac{1}{P}+\frac{1}{R}\right)$$

F1值越高 ⟶ 两者间平衡越好，性能越优

F_β 加权的调跌平均

$$\frac{1}{F_\beta} = \frac{1}{1+\beta^2} \cdot \left(\frac{1}{P} + \frac{\beta^2}{R}\right)$$

if $0 \le \beta^2 < 1$ $\frac{\beta^2}{R}$ 相对小 → P 对 F 的影响大

if $\beta^2 > 1$ $\frac{\beta^2}{R}$ 相对大 → R 对 F 的影响大

F_1 中 P 和 R 的影响是等价的

如果在某些问题中想更看重 P 或 R 可以用 F_β

当进行多次实验时

宏查全率、宏查准率、宏 F_1

假设 K 次实验

$$\begin{matrix} P_1 & R_1 \\ P_2 & R_2 \\ \vdots & \vdots \\ P_k & R_k \\ \bar{P} & \bar{R} \end{matrix}$$

$\bar{P}$ → 宏查准率　　$\bar{R}$ → 宏查全率

$$\frac{1}{F_1} = \frac{1}{2}\left(\frac{1}{\bar{P}} + \frac{1}{\bar{R}}\right)$$

$$macro\text{-}P = \frac{1}{n}\sum_{i=1}^{n} P_i$$

$$macro\text{-}R = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$macro\text{-}F_1 = \frac{2 \times macro\text{-}P \times macro\text{-}R}{macro\text{-}P + macro\text{-}R}$$

微查全率、微查准率、微 F_1

假设 K 次实验

$$\begin{matrix} TP_1 & FP_1 & TN_1 & FN_1 \\ TP_2 & FP_2 & TN_2 & FN_2 \\ \vdots & \vdots & \vdots & \vdots \\ TP_k & FP_k & TN_k & FN_k \\ \overline{TP} & \overline{FP} & \overline{TN} & \overline{FN} \end{matrix}$$

P　R　F_1

$$micro\text{-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}$$

$$micro\text{-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}$$

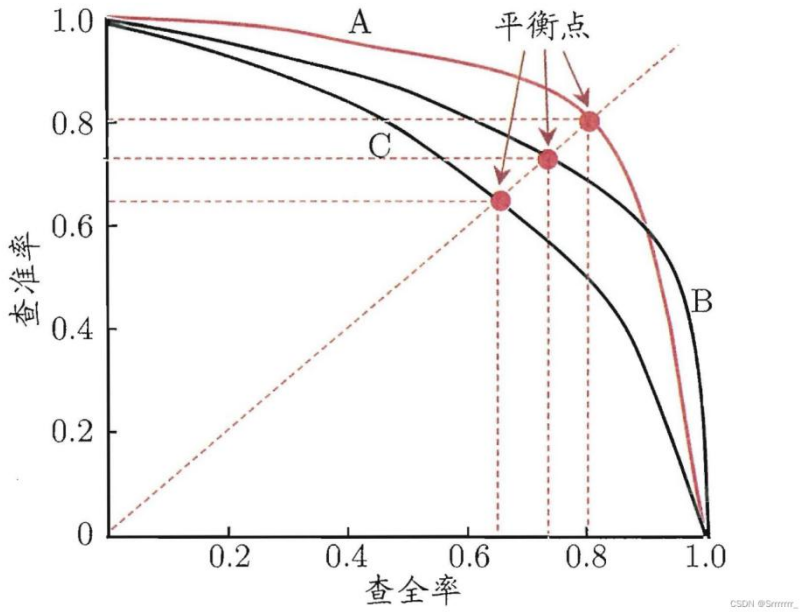
$$micro\text{-}F_1 = \frac{2 \times micro\text{-}P \times micro\text{-}R}{micro\text{-}P + micro\text{-}R}$$

平衡点(BEP)

“平衡点”(Break-Even-Point，简称 BEP)就是这样一个度量，它是“查准率 = 查全率”时的取值

P-R 曲线

在很多情形下,我们可根据学习器的预测结果对样例进行排序,排在前面的是学习器认为“最可能”是正例的样本，排在最后的是学习器认为“最不可能”是正例的样本。
按此顺序设置不同的阈值，逐个把样本作为正例进行预测，则每次可以计算出当前的查准率、查全率。以查准率为纵轴、查全率为横轴作图，就得到了查准率-查全率曲线，简称“P-R 曲线”，显示该曲线的图称为“P-R 图”。



回到刚刚说的平衡点，例如上图中学习器 C 的平衡点是 0.64
而基于平衡点的比较，平衡点越高说明 p 和 r 都有较好的表现，可认为学习器 A 优于 B。

绘制 P-R 曲线

很多学习器是为测试样本产生一个实值或概率预测，然后将这个预测值与一个分类阈值(threshold)进行比较，若大于阈值则分为正类，否则为反类。
例如，很多分类模型可以对每个测试样本预测出一个 [0.0,1.0] 之间的实值，然后将这个值与 0.5 进行比较，大于 0.5 则判为正例，否则为反例。

例　　y = [0, 1, 0, 1, 1]　　1正　0负

预测结果　y_pred = [0.3, 0.9, 0.6, 0.8, 0.7]　(出现率)

假设阈值设成 0.9 → 0 1 0 0 0

0.8 → 0 1 0 1 0

0.7 → 0 1 0 1 1

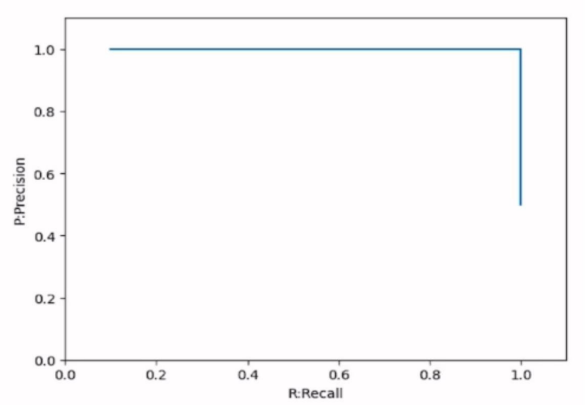
0.6 → 0 1 1 1 1

0.3 → 1 1 1 1 1

} 计算了组 PR

还是上面的例子

	P	R
0.9	100%	$\frac{1}{3}$
0.8	100%	$\frac{2}{3}$ ↑
0.7	100%	100%
0.6	↓	100%
0.5	↓	100%



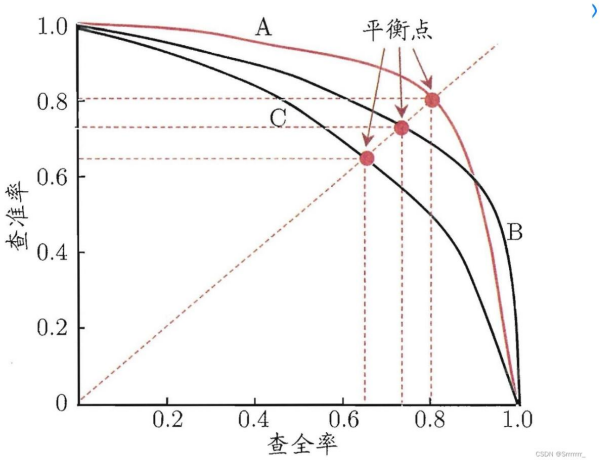
得到这个图

PR曲线通过什么体现模型的好坏

1. C的曲线完全被B曲线包住　则B性能优于C
(R相同时B的P都比C大)

2. 比较P-R曲线下的面积，一定程度上表征了双高的比例
（面积越大，表示模型在P和R之间有更好的权衡，性能越好）

3. 比较平衡点BEP的值，值越大越好



ROC 曲线和 AUC（度量分类中的非均衡性）

ROC 曲线

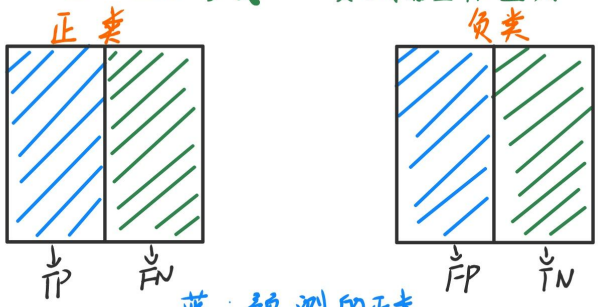
预测值或概率预测结果的好坏，直接决定了学习器的泛化能力。
实际上根据预测值或预测结果，我们可将测试样本进行排序，“最可能”是正例的排在最前面，“最不可能”是正例的排在最后面。这样，分类过程就相当于在这个排序中以某个“截断点”(cut point)将样本分为两部分，前一部分判作正例，后一部分则判作反例。

在不同的应用任务中，我们可以根据任务需求来采用不同的截断点，例如：
更重视“查准率”，则可选择排序中靠前的位置进行截断
更重视“查全率”，则可选择排序中靠后的位置进行截断

因此，排序本身的质量好坏，体现了综合考虑学习器在不同任务下的“期望泛化性能”的好坏，或者说，“一般情况下”泛化性能的好坏。ROC 曲线就是从这个角度出发来研究学习器泛化性能的有力工具。
ROC 曲线和查准率/查全率曲线有着很多的相似之处，当然它们也有所不同。它将真正类率（即recall）和假正类率（被错误分类的负实例）的比例对应着绘制在一张图中，而非使用查准率/查全率。

ROC 曲线

受试者工作曲线



蓝：预测的正类
绿：预测的负类

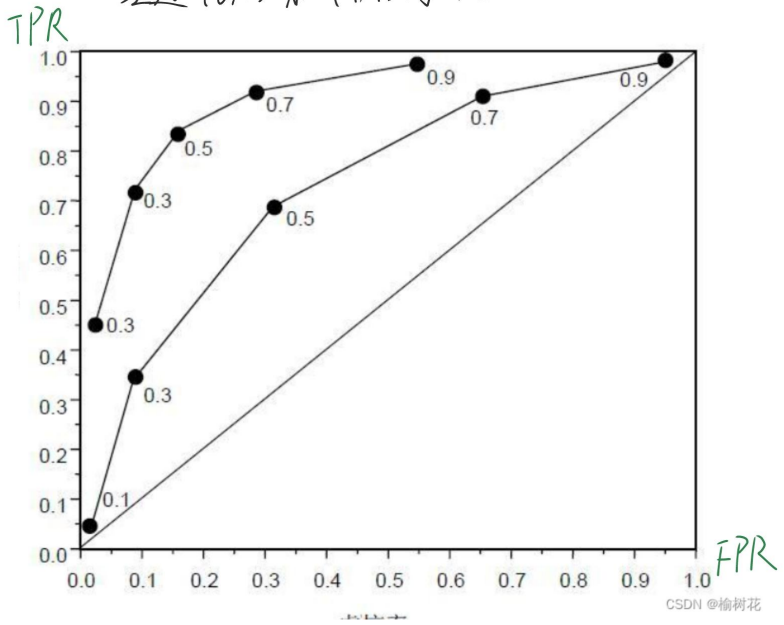
$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}$$

真正例率：正类中预测正确的比例

假正例率：负样本中预测错误的比例

ROC 曲线：以 FPR 为横坐标，TPR 为纵坐标得到的曲线，经过 (0,0) 和 (1,1) 两个点。



1. 曲线越靠左上角性能越好

2. 曲线下面积越大表示在 FPR 和 TPR 之间有更好的权衡，性能越好

横轴 FPR：FPR 越大，预测正类中实际负类越多。

纵轴 TPR：TPR 越大，预测正类中实际正类越多。

理想目标：TPR=1，FPR=0，即图中 (0,1) 点，故 ROC 曲线越靠拢 (0,1) 点，越偏离 45 度对角线越好

相比 P-R 曲线来说，ROC 曲线有一个很大的特点：ROC 曲线的形状不会随着正负样本分布的变化而产生很大的变化，而 P-R 曲线会发生很大的变化。
比如：负样本数量增加一个量级后 P-R 曲线可能会发生特别明显的变化(影响精确率)，而 ROC 曲线形状基本不变。而在实际环境中，正负样本的数量往往是不平衡的，所以 ROC 曲线是最合适的。

AUC 值（Area Unser the Curve）

AUC 是 ROC 曲线下的面积，AUC 值给出的是分类器的平均性能值。使用 AUC 值可以评估二分类问题分类效果的优劣
计算公式如下：

$$AUC = \frac{1+TPR-FPR}{2}$$

一个完美的分类器的 AUC 为 1.0，而随机猜测的 AUC 为 0.5，显然 AUC 值在 0 和 1 之间，并且数值越高，代表模型的性能越好。

简单交叉验证

方法：将原始数据集随机划分成训练集和验证集两部分。

比如说，将样本按照 70%~30%的比例分成两部分，70%的样本用于训练模型；30%的样本用于模型验证。

缺点：（1）数据都只被所用了一次，没有被充分利用
（2）在验证集上计算出来的最后的评估指标与原始分组有很大关系。

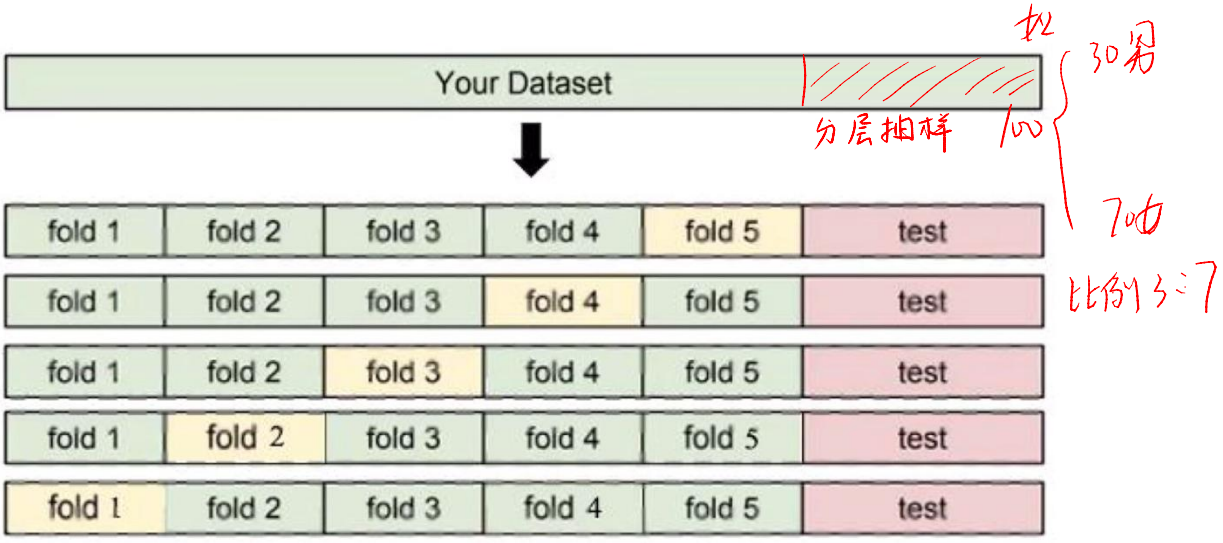
留出法（Hold-out Method）

留出法是直接将数据集 D 划分为两个互斥的集合，其中一个作为训练集 S，另一个作为测试集 T。评估其测试误差，作为对泛化误差的估计。
PS:训练/测试集的划分要尽可能的保持数据分布的一致性，避免因数据划分过程引入额外的偏差而对最终结果产生影响。常用采样方式为“分层采样”。
例如：
数据集 D：1000，训练集：700，测试集：300.
若数据集 D 包含 500 正例，500 反例。
那么采样结果是训练集得有 350 正例，350 反例，测试集有 150 正例 150 反例。

类比：
训练集~模拟试卷，测试集~正式考试试卷。
模拟卷和测试卷考题的类型得分布一样，不然训练就达不到效果。
单次使用留出法得到的估计结果往往不够稳定可靠，在使用留出法时，一般要采用若干次随机划分、重复进行实验评估后取平均值作为留出法的评估结果。

k-折交叉验证（k-Fold Cross Validation）

1、首先，将全部样本划分成 k 个大小相等的样本子集；
2、依次遍历这 k 个子集，每次把当前子集作为验证集，其余所有样本作为训练集，进行模型的训练和评估；
3、最后把 k 次评估指标的平均值作为最终的评估指标。
比较不同模型的性能，选择最优训练集和验证集训练模型，超参数调优，防止过拟合，评估模型的稳定性

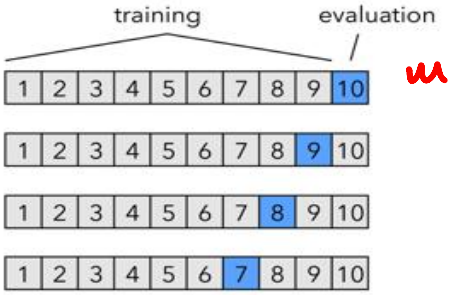


则每一个fold里的男女比都要为3:7
避免因数据划分引入额外的偏差

fold1…6 训练 模型f1 → folds 验证 → 误差e1
↓以此类推
得到e1 e2 e3 e4 e5 求平均误差ē

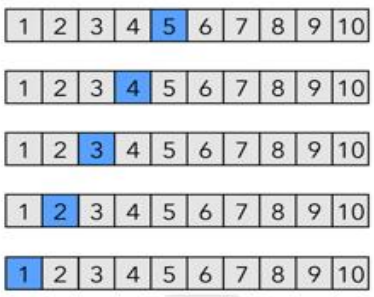
留一验证（LOOCV，Leave one out cross validation ）

只从可用的数据集中保留一个数据点，并根据其余数据训练模型。此过程对每个数据点进行迭代，比如有 n 个数据点，就要重复交叉验证 n 次。例如下图，一共 10 个数据，就交叉验证十次



m个

K折 → 每个fold → m/K
m折 → 每个fold → 1
每次m-1个 训练
1个 验证



优点：
适合小样本数据集
利用所有的数据点，因此偏差将很低
缺点：
重复交叉验证过程 n 次导致更高的执行时间
测试模型有效性的变化大。因为针对一个数据点进行测试，模型的估计值受到数据点的很大影响。如果数据点被证明是一个离群值，它可能导致更大的变化