对于模型来说，在训练集上的误差称为训练误差（training error）或经验误差（empirical error），测试集的误差称为测试误差（testing error）或者泛化误差（generalization error）。

也就是说测试误差是用来评估模型对于新样本的学习能力，因此我们更关注的是测试误差，我们希望模型可以从现有的数据中学习到**普遍规律**而用于新样本。

因此我们需要将现有数据集（data set）划分为训练集（trainning set）和测试集（test set），其中训练集用来训练模型，而测试集用来验证模型对新样本的判别能力。那么，划分数据集有什么做法呢？

1. 留出法（hold-out）

直接将数据集D划分为两个互斥的集合：训练集S和测试集T（D = S∪T，S∩T = ∅），在S上训练模型，用T来评估其测试误差。

要点：

1. 训练/测试集的划分要尽可能**保持数据分布的一致性（进行分层划分stratified sampling）**，避免因为数据划分过程引入额外的偏差而对最终结果产生影响。例如：在分类问题中，S与T中正负样本的比例应该接近。
2. 单次使用留出法得到的估计往往不够稳定可靠，为得到合理的算法模型评估标准，应该**进行多次数据集的随机划分**，取**评估结果平均值**作为评判模型优劣的标准。
3. 训练集太大，评估的结果可能不太准确稳定。训练集太小，会产生较大的偏差（high bias）。因此，实际中一般将大约2/3～4/5的样本用于训练（推荐S:T=7:3），剩余样本用于测试。

import pandas as pd

import os

from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

os.chdir(r'C:\Users\o\Desktop\breast\_cancer\_data')

data = pd.read\_csv(r'data.csv')

data.drop('Unnamed: 32',inplace = True,axis = 1)

data.drop('id',inplace = True,axis=1)

y = data['diagnosis']

x = data.drop('diagnosis',axis = 1)

model = RandomForestClassifier()

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.33, random\_state=42)

print(x\_train.shape)

print(x\_test.shape)

model.fit(x\_train,y\_train)

prediction = model.predict(x\_test)

metrics.accuracy\_score(prediction,y\_test)

0.925531914893617

二. 交叉验证（cross validation）

在NG（吴恩达）的ML课程有提到：最佳的数据分类情况应当把数据集分为三个部分：训练集S、测试集T以及交叉验证集(cross validation set)CV，比例为S:T:CV=0.6：0.2:0.2。我们应当在S上训练模型，用CV**代替**T根据评价标准来**选择模型。**

**实际上，CV与T的主要区别是：验证集用于进一步确定模型中的超参数(例如正则项系数、ANN中隐含层的节点个数等)而测试集只是用于评估模型的精确度(即泛化能力)！**

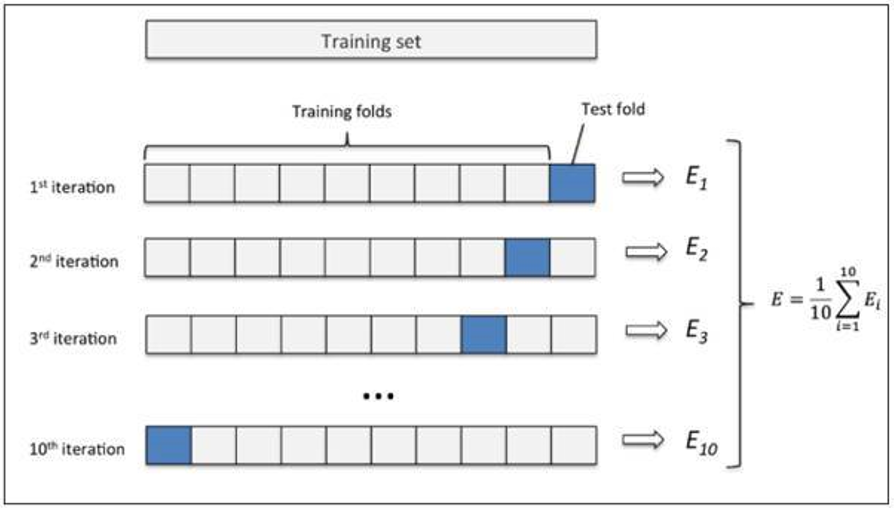
首先要明确的一点是，交叉验证是用于**模型选取(model selection)**过程中的，而不是**模型训练**过程中的！

三．K折交叉验证K-FOLD CROSS VALIDATION

先将数据集D划分为k（一般而言k的取值为10，常用的还有5、20等

）个大小相似的**互斥**子集.每个子集Di都尽可能**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，余下的那个子集作为测试集**，这样就可以获得k组训练/测试集，从而可以进行k次训练和测试，最终返回的是这**k个测试结果的均值**。



**Sklearn中K折交叉验证函数有：KFold，GroupKFold，StratifiedKFold，本篇介绍KFold**

Import numpy as np

from sklearn. model\_selection import KFold

kf = KFold(n\_splits = 10)

accuracy = [ ]

for train\_index,test\_index in kf.split(x):

x\_train,x\_test = x.loc[train\_index],x.loc[test\_index]

y\_train,y\_test = y.loc[train\_index],y.loc[test\_index]

model.fit(x\_train,y\_train)

prediction = model.predict(x\_test)

acc=metrics.accuracy\_score(prediction,y\_test)

accuracy.append(acc)

print(accuracy)

print(np.average(accuracy))

1#output

2[0.9122807017543859, 0.8771929824561403, 0.9649122807017544, 0.9824561403508771, 0.9649122807017544, 0.9649122807017544, 0.9649122807017544, 0.9649122807017544, 0.9649122807017544, 1.0]

30.956140350877193

四．留一法LOO& LPO

LOO留一法是**特殊的K折交叉验证**，即k=m（m为样本总数量），就是说每次进行只留下一个样本用作测试集，其余m-1全为训练集，进行m次训练，取m次的评估结果的平均值进行模型选择。

LPO留P法是指：有m个样本，将每P个样本作为测试样本，其它m-P个样本作为训练样本，这样得到c(p,m)=m!/((m-p)!\*p!)个train-test pairs，当P>1时，测试集将会发生重叠，当P=1的时候，就变成了留一法。

LOO的好处在于,不受样本随机划分带来的偏差，往往被认为比较准确，避免的数据的浪费,但是同时也拥有了**更高的计算开销** 。

一般LOO相对于 K-Fold 而言,拥有更高的方差,但是对于方差占主导的情况时,LOO可能拥有比交叉验证更强的能力.

from sklearn.model\_selection import LeaveOneOut,LeavePOut

loo = LeaveOneOut()

accuracy = []

for train\_index,test\_index in loo.split(x):

x\_train,y\_train,x\_test,y\_test = x.loc[train\_index],y.loc[train\_index],x.loc[test\_index],y.loc[test\_index] model.fit(x\_train,y\_train)

prediction = model.predict(x\_test)

acc=metrics.accuracy\_score(prediction,y\_test)

accuracy.append(acc)

print(np.average(accuracy))

五．自助法bootstraping

"Pull oneself up by one's bootstrap"

Bootstrap来源于这句很有趣的话，字面意思是拎着鞋带把自己提起来，翻译为自力更生。



我们希望评估的是用D训练出来的模型，但是留出法和交叉验证法中，由于保留了一部分样本用于测试，因此**实际评估的模型所使用的训练集比D小**，这必然会引入一些因训练样本规模不同而导致的估计偏差，为此提出自助法。

给定包含m个样本的数据集D，我们对它进行采样产生数据集 D′：每次随机从D中挑选出一个样本，将其拷贝放入D′, 然后再将该样本**放回**初始数据集D中，使得该样本在下次采样时仍有可能被采样到；这个过程重复执行m次后，我们就得到可包含m个样本数据的数据集D′,这就是自助采样的结果.

样本在m次采样中始终不被采到到概率为

https://gss0.bdstatic.com/-4o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D198/sign=0b2f21e28826cffc6d2abbbb81004a7d/63d9f2d3572c11df1586e858692762d0f703c272.jpg

由此可知通过自助采样，初始数据集D中约有36.8%的样本未出现在采样数据集D′中。于是我们可将D′ 用作训练集，D∖D′(\为集合减法)用作测试集。

优缺点：自助法在**数据集较小**，难以有效划分训练/测试集时很有用，但是，**自助法改变了初始数据集的分布**，这会引入估计偏差，所以在数据量足够时，一般采用留出法和交叉验证法。[4]

六．随机划分法

Sklearn中的函数有：ShuffleSplit，GroupShuffleSplit，StratifiedShuffleSplit，本文使用shufflesplit

ShuffleSplit迭代器产生指定**数量的独立**的train/test数据集划分，首先对样本全体随机打乱，然后再划分出train/test对，可以使用随机数种子random\_state来控制数字序列发生器使得讯算结果可重现

ShuffleSplit是KFlod交叉验证的比较好的替代，他允许更好的**控制迭代次数和train/test的样本比例**

from sklearn.model\_selection import ShuffleSplit

rs = ShuffleSplit(n\_splits = 10,test\_size = 0.3)

accuracy = []

for train\_index,test\_index in rs.split(x):

x\_train,y\_train,x\_test,y\_test = x.loc[train\_index],y.loc[train\_index],x.loc[test\_index],y.loc[test\_index]

model.fit(x\_train,y\_train)

prediction = model.predict(x\_test)

acc=metrics.accuracy\_score(prediction,y\_test)

accuracy.append(acc)

print(np.average(accuracy))

参考资料：

百度词条《抽样数据方式》

周志华《机器学习》

《Patttern Recognition and Machine Learning》

Sklearn API文档

吴恩达《Machine Learning》公开课