**scikit-learn——交叉验证**

**任务目标**

1.了解交叉验证的各种方法

2.熟练掌握sklearn.cross\_validation(升级后修改为sklearn.model\_selection)中的交叉验证方法的使用

**相关知识**

sklearn是利用python进行机器学习中一个非常全面和好用的第三方库，sklearn中关于交叉验证有很多种用法，在sklearn.cross\_validation中包含：

①KFold：K-Fold交叉验证迭代器。接收元素个数、fold数、是否清洗

②LeaveOneOut：LeaveOneOut交叉验证迭代器

③LeavePOut：LeavePOut交叉验证迭代器

④LeaveOneLableOut：LeaveOneLableOut交叉验证迭代器

⑤LeavePLabelOut：LeavePLabelOut交叉验证迭代器

下面是每个交叉验证的具体用法：

KFold交叉验证：sklearn.model\_selection.KFold(n\_splits=3, shuffle=False, random\_state=None)

思路：将训练/测试数据集划分n\_splits个互斥子集，每次用其中一个子集当作验证集，剩下的n\_splits-1个作为训练集，进行n\_splits次训练和测试，得到n\_splits个结果

注意点：对于不能均等分的数据集，其前n\_samples % n\_splits子集拥有n\_samples //n\_splits + 1个样本，其余子集都只有n\_samples / n\_splits样本

参数说明：

n\_splits：表示划分几等份

shuffle：在每次划分时，是否进行洗牌

      ①若为False时，其效果等同于random\_state等于整数，每次划分的结果相同

      ②若为True时，每次划分的结果都不一样，表示经过洗牌，随机取样的

random\_state：随机种子数

属性：

     ①get\_n\_splits(X=None, y=None, groups=None)：获取参数n\_splits的值

     ②split(X, y=None, groups=None)：将数据集划分成训练集和测试集，返回索引生成器

留一交叉验证 (LOO)：

留一交叉验证是一种用来训练和测试分类器的方法，会用到图像数据集里所有的数据，假定数据集有N个样本（N1、N2、...Nn），将这个样本分为两份，第一份N-1个样本用来训练分类器，另一份1个样本用来测试，如此从N1到Nn迭代N次，所有的样本里所有对象都经历了测试和训练。

随机排列交叉抽样ShuffleSplit：

sklearn.model\_selection.ShuffleSplit(n\_splits=10, test\_size=’default’, train\_size=None, random\_state=None)

数据集在进行划分之前，首先是需要进行打乱操作，否则容易产生过拟合，模型泛化能力下降。

参数说明：

n\_splits: int (default 10)：重新洗牌和分裂迭代次数。

test\_size : float (default 0.1), int, or None：如果是float类型的数据, 这个数应该介于0-1.0之间，代表test集所占比例. 如果是int类型, 代表test集的数量. 如果为None, 值将自动设置为train集大小的补集

train\_size : float, int, or None (default is None)：如果是float类型的数据 应该介于0和1之间，并表示数据集在train集分割中所占的比例 如果是int类型, 代表train集的样本数量. 如果为None, 值将自动设置为test集大小的补集

random\_state : int or RandomState：用于随机抽样的伪随机数发生器状态。

**系统环境**

Python

Jupyter

**任务内容**

练习sklearn中交叉验证的各种方法的使用。

**任务步骤**

1.首先打开终端模拟器，输入下面命令，更新scikit-learn库。

1. conda update scikit-learn
2. 或者：
3. pip -U scikit-learn

2.打开终端模拟器，在命令行输入ipython notebook --ip='127.0.0.1'，在浏览器中会打开下面界面，点击New

3.新建一个ipynb文件，用于编写并执行代码。

交叉验证迭代器

4.K折交叉验证（KFold）：将样例划分为K份，若K=len(样例)，即为留一交叉验证，K-1份作为训练，剩余样例作为测试集，下面是一个例子，在有4个样本的数据集上进行2折交叉验证。

首先导入numpy模块，别名为np，导入sklearn.model\_selection中的KFold类。

1. **import** numpy as np
2. from sklearn.model\_selection **import** KFold

然后创建由"a", "b", "c", "d"组成的列表X，再使用KFold类，传入参数n\_splits=2创建2-fold对象kf，最后调用kf中的split方法，对X进行2折交叉验证迭代，并用for循坏遍历2折交叉验证迭代结果，每一个折叠都由两个数组组成：第一个是与训练集相关的，第二个是与测试集相关的。

1. X = ["a", "b", "c", "d"]
2. kf = KFold(n\_splits=2)
3. **for** train\_index, test\_index **in** kf.split(X):
4. print("%s %s" % (train\_index, test\_index))



5.重复 K-折交叉验证（RepeatedKFold）：重复 K-折交叉验证n次，每次重复产生不同的分裂，下面是一个例子，在有4个样本的数据集上进行重复2次的2折交叉验证。

首先，导入numpy模块，别名为np，导入sklearn.model\_selection中的RepeatedKFold。

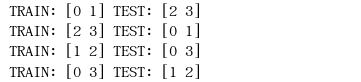
1. **import** numpy as np
2. from sklearn.model\_selection **import** RepeatedKFold

然后，使用np.array创建二维数组X，一维数组y，

1. X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
2. y = np.array([0, 0, 1, 1])

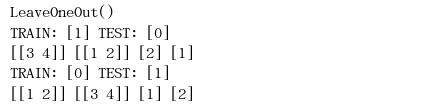
最后，使用RepeatedKFold类，传入参数n\_splits=2, n\_repeats=2, random\_state=2652124，创建重复2次的2-fold对象rkf，调用rkf中的split方法，对X进行折叠，并用for循坏遍历折叠结果，每一个折叠都由两个数组组成：第一个是与训练集相关的，第二个是与测试集相关的。

1. rkf = RepeatedKFold(n\_splits=2, n\_repeats=2, random\_state=2652124)
2. **for** train\_index, test\_index **in** rkf.split(X):
3. print("TRAIN:", train\_index, "TEST:", test\_index)
4. X\_train, X\_test = X[train\_index], X[test\_index]
5. y\_train, y\_test = y[train\_index], y[test\_index]



6.留一交叉验证 (LOO)：其实相当于KFold(n\_splits=n)，或LeavePOut(p=1)，这里的n是样本数。下面是一个例子，在有2个样本的数据集上进行留一交叉验证。

1. **import** numpy as np
2. from sklearn.model\_selection **import** LeaveOneOut
3. X = np.array([[1, 2], [3, 4]])
4. y = np.array([1, 2])
5. loo = LeaveOneOut()
6. loo.get\_n\_splits(X)
8. print(loo)
10. **for** train\_index, test\_index **in** loo.split(X):
11. print("TRAIN:", train\_index, "TEST:", test\_index)
12. X\_train, X\_test = X[train\_index], X[test\_index]
13. y\_train, y\_test = y[train\_index], y[test\_index]
14. print(X\_train, X\_test, y\_train, y\_test)



7.留P交叉验证（LeavePOut）： 是从数据集中随机的选取p个样本作为测试集，剩下的样本作为训练集，重复抽样，直到把所有结果都取到，当p>1时，测试集数据有重叠，下面是一个留P交叉验证例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** LeavePOut
3. X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
4. y = np.array([1, 2, 3, 4])
5. lpo = LeavePOut(1)
6. lpo.get\_n\_splits(X)
7. print(lpo)
8. **for** train\_index, test\_index **in** lpo.split(X):
9. print("TRAIN:", train\_index, "TEST:", test\_index)
10. X\_train, X\_test = X[train\_index], X[test\_index]
11. y\_train, y\_test = y[train\_index], y[test\_index]

[](https://www.ipieuvre.com/doc/exper/368c5599-91ad-11e9-beeb-00215ec892f4/img/21cc9a1f-acf7-4372-bdd4-114c62afa29b.png)

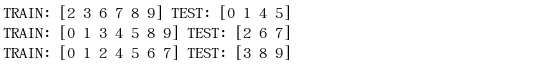
8.随机排列交叉抽样（ShuffleSplit ）：首先将样本随机打乱，然后根据设置参数划分训练数据集与测试数据集，其中参数n\_splits：设置重新洗牌和分裂迭代次数，参数test\_size=0.25：设置测试集的比例，参数random\_state：设置随机抽样的状态。下面是一个随机排列交叉抽样的例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** ShuffleSplit
3. X = np.arange(5)
4. ss = ShuffleSplit(n\_splits=3, test\_size=0.25,
5. random\_state=0)
6. **for** train\_index, test\_index **in** ss.split(X):
7. print("%s %s" % (train\_index, test\_index))



9.分层k折（StratifiedKFold）： 是 k-fold 的变种，会返回分层的折叠，即每个小集合中， 各个类别的样例比例大致和完整数据集中相同，通过指定分组，对测试集进行无放回抽样，下面是一个分层k折交叉抽样的例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** StratifiedKFold
4. X = np.ones(10)
5. labels = [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
6. skf = StratifiedKFold(n\_splits=3)
7. **for** train\_index, test\_index **in** skf.split(X, labels):
8. print("TRAIN:", train\_index, "TEST:", test\_index)



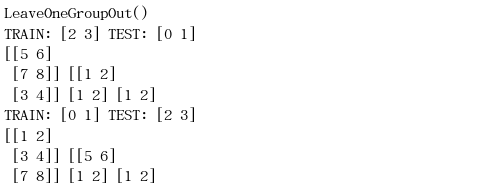
10.分组K折交叉验证（GroupKFold）：先分组，然后把所有组划分为K份，随机取K-1份作为训练，剩余一份作为测试集，这里K 小于分组的组数，下面是一个分组k折交叉抽样的例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** GroupKFold
3. X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
4. y = np.array([1, 2, 3, 4])
5. groups = np.array([0, 0, 2, 2])
6. group\_kfold = GroupKFold(n\_splits=2)
7. print(group\_kfold)
8. **for** train\_index, test\_index **in** group\_kfold.split(X, y, groups):
9. print("TRAIN:", train\_index, "TEST:", test\_index)
10. X\_train, X\_test = X[train\_index], X[test\_index]
11. y\_train, y\_test = y[train\_index], y[test\_index]
12. print(X\_train, X\_test, y\_train, y\_test)



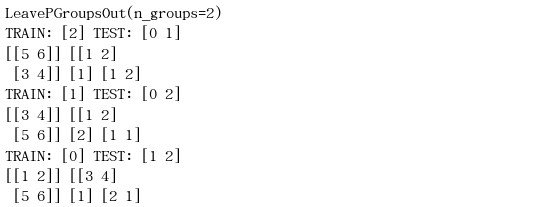
11.分组留一交叉验证LeaveOneGroupOut：先分组，然后随机取一组做测试集，剩下组做训练数据集，这个是在GroupKFold 的基础上混乱度又减小了，下面是一个分组k折交叉抽样的例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** LeaveOneGroupOut
3. X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
4. y = np.array([1, 2, 1, 2])
5. groups = np.array([1, 1, 2, 2])
6. logo = LeaveOneGroupOut()
7. logo.get\_n\_splits(X, y, groups)
8. print(logo)
9. **for** train\_index, test\_index **in** logo.split(X, y, groups):
10. print("TRAIN:", train\_index, "TEST:", test\_index)
11. X\_train, X\_test = X[train\_index], X[test\_index]
12. y\_train, y\_test = y[train\_index], y[test\_index]
13. print(X\_train, X\_test, y\_train, y\_test)



12.分组留P交叉验证LeavePGroupsOut：先分组，然后随机取P组做测试集，剩下组做训练数据集，下面是一个分组留P交叉验证抽样的例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** LeavePGroupsOut
3. X = np.array([[1, 2], [3, 4], [5, 6]])
4. y = np.array([1, 2, 1])
5. groups = np.array([1, 2, 3])
6. lpgo = LeavePGroupsOut(n\_groups=2)
7. lpgo.get\_n\_splits(X, y, groups)
9. lpgo.get\_n\_splits(groups=groups)  # 'groups' is always required
11. print(lpgo)
13. **for** train\_index, test\_index **in** lpgo.split(X, y, groups):
14. print("TRAIN:", train\_index, "TEST:", test\_index)
15. X\_train, X\_test = X[train\_index], X[test\_index]
16. y\_train, y\_test = y[train\_index], y[test\_index]
17. print(X\_train, X\_test, y\_train, y\_test)



13.分组随机排序交叉验证GroupShuffleSplit：先分组，然后将组排序随机打乱，最后根据设置参数划分训练数据集与测试数据集，下面是一个分组随机排序交叉验证的例子。

1. **import** numpy as np
2. from sklearn.model\_selection **import** GroupShuffleSplit
3. X = [0.1, 0.2, 2.2, 2.4, 2.3, 4.55, 5.8, 0.001]
4. y = ["a", "b", "b", "b", "c", "c", "c", "a"]
5. groups = [1, 1, 2, 2, 3, 3, 4, 4]
6. gss = GroupShuffleSplit(n\_splits=4, test\_size=0.5, random\_state=0)
7. **for** train\_index, test\_index **in** gss.split(X, y, groups=groups):
8. print("TRAIN:", train\_index, "TEST:", test\_index)

