机器学习应用实践（实验五）——集成学习之随机森林

一、实验目的

1、熟悉随机森林算法的原理。

2、掌握 Scikit-Learn 中随机森林 API 的使用，理解算法中各参数的含义和作用。

3、掌握 Scikit-Learn 中提供的调参工具随机网格搜索和对半网格搜索的使用。

4、具备使用 python 实现随机森林算法的编程能力。

二、实验准备

1、回顾随机森林算法的原理。

2、学习 Scikit-Learn 中随机森林算法涉及到的相关知识。

三、相关知识介绍

1、集成学习概述

集成学习(ensemble learning)是目前机器学习中最先进、最有效、最具研究价值的领域之一，集成学习本身不是一个单独的机器学习算法，而是通过在数据上构建多个模型，并将他们的输出结果以某种方式结合起来解决一个问题。组成集成模型的每个模型叫做个体学习器，如果用来集成的模型都是同一种类型（“同质”）的个体学习器，这样的个体学习器叫做基学习器(base learner)，如果用来集成的模型不是同一种类型（“异质”）的个体学习器，这样的个体学习器叫做组件学习器(component learner)。个体学习器可以是弱学习器或强学习器。

根据个体学习器构造的策略不同和个体学习器输出的集成策略不同，集成学习可以被分为三个主要研究领域：

（1）模型融合

模型融合的个体学习器主要是强学习器，个体学习器的输出集成方法主要包括投票法 Voting、学习法 Stacking、混合法 Blending 等。模型融合技巧是机器学习/深度学习竞赛中最为可靠的提分手段之一。

（2）弱学习器集成

将多个弱学习器（指泛化性能略优于随机猜测的学习器）集成得到一个强学习器。这个领域涵盖了大部分我们熟悉的集成算法和集成手段，如装袋法 bagging，提升法 boosting 等，随机森林、Xgboost 都属于这个领域。

（3）混合专家模型(mixture of experts)

混合专家模型常常出现在深度学习(神经网络)的领域，在其他集成领域当中，不同的学习器是针对同一任务、甚至在同一数据上进行训练，但在混合专家模型中，是将一个复杂的任务拆解成几个相对简单且更小的子任务，然后针对不同的子任务训练个体学习器(专家)，然后再结合这些个体学习器的结果得出最终的输出。

2、Bagging 方法的基本思想

Bagging 又称为“装袋法”，它是所有集成学习方法当中非常简单但又十分有效的方法。它的策略就是并行建立多个弱评估器(通常是决策树，也可以是其他非线性算法)，多个弱评估器之间是相互独立的，并综合多个弱评估器的结果进行输出。当集成算法目标是回归任务时，集成算法的输出结果是弱评估器输出的结果的平均值；当集成算法的目标是分类任务时，集成算法的输出结果是弱评估器输出的结果少数服从多数（采用投票方式），如图 1 所示。

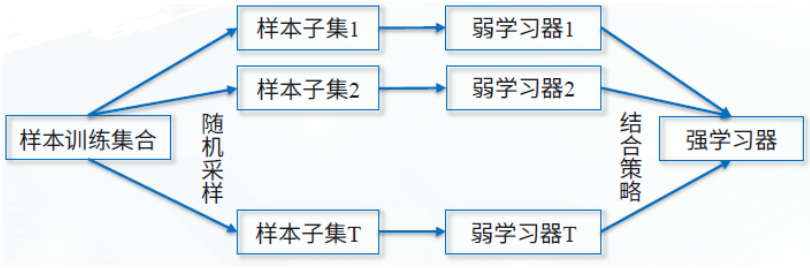

图 1 Bagging 策略

在 sklearn 中，有两个 Bagging 集成算法：随机森林(RandomForest)和极端随机树(ExtraTrees)，具体见表 1，他们都是以决策树为弱评估器的有监督算法，可以被用于分类、回归、排序等各种任务。同时，还可以使用 Bagging 的思想对其他算法进行集成，比如使用装袋法分类算法 BaggingClassifier 对任意（决策树之外，如支持向量机或逻辑回归）的算法进行集成，下面将重点介绍随机森林的原理与用法。

表 1　sklearn.ensemble 中的 Bagging 集成算法

Bagging 集成算法	**sklearn.ensemble 中的类**
随机森林分类	RandmForestClassifier
随机森林回归	RandomForestRegressor
极端随机树分类	ExtraTreesClassifier
极端随机树回归	ExtraTreesRegressor
装袋法分类	BaggingClassifier
装袋法回归	BaggingRegressor

3、Scikit-Learn 中随机森林算法概述

随机森林是 Bagging 策略的代表算法，它的所有基评估器都是决策树。随机森林算法的基本流程是：从提供的数据中有放回抽样（比如 a bootstrap sample 自助式采样法）出不同的子集，使用这些子集并行建立多棵不同的决策树，并按照 Bagging 的规则(回归则平均，分类则投票)对单棵决策树的结果进行集成。

虽然原理上很简单，但很多学习任务中展现出强大的性能，被誉为“代表集成学习技术水平的方法”。在机器学习竞赛当中，随机森林往往是在中小型数据上会尝试的第一个算法。

在 sklearn 中，随机森林回归器由类 sklearn.ensemble.RandomForestRegressor 实现，随机森林分类器则由类 sklearn.ensemble.RandomForestClassifier 实现。我们可以像调用逻辑回归、决策树等其他

sklearn 中的算法一样，使用“实例化、fit、predict/score”三部曲来使用随机森林，同时我们也可以使用 sklearn 中的交叉验证方法来实现随机森林。其中回归森林的默认评估指标为 R^2，分类森林的默认评估指标为分类准确率。随机森林回归器和分类器的参数非常相似，掌握其中一个类的参数就可以了。

3.1 随机森林分类器 RandomForestClassifier

与其他分类器一样，由树分类器组合成的“森林”分类器需要拟合两个数组：一个是训练样本数组 X（n_samples, n_features）（可以是稀疏数组或非稀疏数组），另一个是训练样本的目标值（类标签）数组 Y (n_samples,)。

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```

与决策树一样，“森林”分类器也可以扩展用于多输出问题，只需要将 Y 变为一个对应的数组 (n_samples, n_outputs)。

sklearn 中随机森林的实现是通过对分类器的概率预测进行平均来组合分类器，而不是让每个分类器为单个类别进行投票。官网地址：https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier。

class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, bootstrap=True, random_state=None, min_samples_split=2)

（1）重要参数

●n_estimators：int，default = 100。随机森林中决策树的数量，即基评估器的数量。这个参数对随机森林模型的复杂度影响是单调的，n_estimators 越大，模型越复杂，模型在训练集效果往往越好。但是 n_estimators 达到一定的程度之后，随机森林训练集分数会达到上限。n_estimators 越大，需要的计算量和内存也越大，训练时间越长。对于这个参数的确定，需要在训练难度和模型效果之间取得平衡。

●random_state：int, default=None。随机森林的随机性种子。通过下面的示例程序可以看到，当随机森林的随机数种子确定时，拟合的决策树集合中的树都以不同的随机性选择最优分枝。我们可以通过 estimators_属性，查看随机森林中决策树的参数。

```
rfc = RandomForestClassifier(n_estimators=5,random_state=10)
rfc = rfc.fit(Xtrain,Ytrain)

rfc.estimators_
```

```
[DecisionTreeClassifier(max_features='auto', random_state=1165313289),
 DecisionTreeClassifier(max_features='auto', random_state=1283169405),
 DecisionTreeClassifier(max_features='auto', random_state=89128932),
 DecisionTreeClassifier(max_features='auto', random_state=2124247567),
 DecisionTreeClassifier(max_features='auto', random_state=574014784)]
```

sklearn 中随机森林的随机性在于三点：

1）每颗决策树的使用完整训练集的一个随机子集生成(bootstrap=True，max_samples)。

2）每颗决策树的节点分枝使用完整特征集的一个随机子集进行(max_features<n_features)。

3）即使训练数据相同(bootstrap=False)，在完整特征集中寻找最优分枝。(max_features=n_features)，在搜索最佳分枝的过程中，如果枚举的几个分枝对准则的改进相同，那么找到的最佳分枝也会有所不同。

需要注意的是，随机森林算法在总数据量有限的情况下，单棵树使用的数据量越大，每一棵树使用的数据就会越相似，每棵树的结构也就会越相似，bagging 的效果难以发挥，模型也很容易过拟合。因此，当数据量足够时，我们往往会消减单棵树使用的数据量。

●bootstrap：bool, default=True。袋装法是通过有放回的随机抽样技术来形成不同的训练数据，bootstrap 就是用来控制抽样技术的参数。为 True 时表示构建树时采用自助集(bootstrap samples)。如果为 False，则使用整个数据集来构建每个树。

由于是有放回的，一些样本可能在同一个自助集中出现多次，一些样本却可能永远不出现，在样本量为 N 的数据集中，一些样本永远不会被抽到的概率为近似为 0.368（当样本量趋于无穷大时），公式如下：

$$\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^{N}=\frac{1}{e}\approx 0.368$$

因此，会有约 37%的训练数据被浪费掉，没有参与建模，这些数据被称为袋外数据(out of bag data，简写为 oob)。除了最开始就划分好的测试集之外，这些数据也可以被用来作为集成算法的测试集，也就是说，在使用随机森林时，可以不划分训练集和测试集，只需要用袋外数据测试模型即可。

●max_samples：int or float, default=None。通常与 bootstrap=True 共同控制从训练样本数组 X 中抽取的样本量来训练每个基估计器。如果 bootstrap=True，max_samples 为 None(默认)，则抽取 X.shape[0](与训练样本个数相同的)样本。max_samples 为 int 时，则抽取 max_samples 个样本。max_samples 为 float 时，则抽取 max_samples * X.shape[0]样本，此时 max_samples 应该在区间(0,1)内。

●oob_score：bool, default=False。是否使用袋外样本来估计泛化分数。仅当 bootstrap=True 时才可用。oob_score 为 True 表示希望使用袋外数据进行测试。训练完成之后，可以使用 oob_score_属性查看袋外数据的测试分数，评价指标默认为 R^2 且不可更改。

程序示例：

```
# 使用袋外数据进行测试
rfc = RandomForestClassifier(oob_score=True)
rfc = rfc.fit(wine.data,wine.target)

rfc.oob_score_
```

```
0.9887640449438202
```

其他控制基评估器（决策树）参数如表 2 所示：

表 2 决策树参数表

参数	含义
criterion	不纯度的衡量指标，有基尼系数和信息熵两种选择，选择信息熵时 sklearn 默认通过父节点和子节点的信息增益进行分枝
max_depth	树的最大深度，超过最大深度的树枝都会被剪掉
min_samples_leaf	叶子节点的最小样本数，一个节点在分枝后的每个子节点都必须包含至少 min_samples_leaf 个训练样本，否则分枝就不会发生
min_samples_split	节点划分最少样本数，一个节点必须包含至少 min_samples_split 个训练样本，这个节点才允许被分枝，否则分枝就不会发生
max_features	每个决策树的最大特征数量，超过 max_features 个数的特征都会被舍弃，默认值为总特征个数开平方取整
max_leaf_nodes	决策树最多可以有的叶子节点数量
min_impurity_decrease	限制信息增益的大小，信息增益小于设定数据的分枝不会发生
min_weight_fraction_leaf	当样本权重(class_weight 设置)被调整时，叶子节点上最少要拥有的基于样本权重的样本量
class_weight	类别权重，当样本标签不均衡时，用于平衡每个类别的权重的参数。
ccp_alpha	结构风险权重。在 sklearn 的 0.22 版本中才加入的参数，在决策树损失函数上加一个结构风险项，该参数就是结构风险项的权重系数。

●warm_start：主要用于增量学习[1]。如果 warm_start 为 true，当前模型的训练结果会保存下来，下一次仍使用该模型进行训练时，不会擦除之前的训练结果，而是使用之前的结果作为初始值，在前一阶段的训练结果上继续训练。否则，warm_start 为 False 时，每次训练都会擦除上一次的训练模型，这也是 sklearn 中的默认设置，这样交叉验证时才有效。

sklearn 中许多评估器（集成模型、线性模型等）都具有 warm_start 参数，在随机森林中，使用 warm_start 可以保留之前建过的树，在之后森林中添加更多的树（增加 n_estimator）。

（2）重要属性

随机森林的大部分属性与决策树相同，也同样可以查看森林中的单颗树的 tree_属性，获取更多树信息。个别不同属性如下：

●estimators_：拟合的决策树列表。

●oob_score_：使用袋外估计获得的测试分数（只有当 oob_score =True 时）。

（3）重要方法

随机森林提供的方法比决策树少些获取单颗决策树信息的方法（如 get_depth，get_n_leaves 等）。

方法	说明
apply(X[, check_input])	返回森林中叶子节点上被预测样本 X 的索引。

[1] 注释：增量学习是指一个学习系统能够不断地从新样本中学习新的知识，并能够保存大部分已经学到的知识。它主要关注的是灾难性遗忘（catastrophic forgetting），平衡新知识与旧知识之间的关系，即如何在学习新知识的情况下不忘记旧知识。

方法	说明
decision_path(X[, check_input])	返回森林中的决策路径。
fit(X, y[, sample_weight, check_input, …])	根据训练集（X，y）建立森林分类器。
get_params([deep])	获取此估算器的参数。
predict(X[, check_input])	预测 X 的类别或回归值。
predict_log_proba(X)	预测输入样本 X 的类对数概率。
predict_proba(X[, check_input])	预测输入样本 X 的类别概率。
score(X, y[, sample_weight])	返回给定测试数据和标签上的平均准确度。
set_params(**params)	设置此估算器的参数。

3.2 随机森林回归器 RandomForestRegressor

RandomForestRegressor 的参数与 RandomForestClassifier 基本一致，只有特征选择标准 criterion 不同。

●criterion：特征选择标准。包括：’squared_error’, ’ absolute_error’, ’poisson’，默认为’squared_error’。

4、随机森林算法流程与实现

- 假设：样本数量为 N，样本属性数量为 d。
- 流程

（1）Bootstrap 法抽 N 个样本形成第 t 个样本子集 Dt={Xt，Yt}。

（2）将 Dt 作为当前决策树 Trt 的训练样本集。

（3）随机选取 m 个属性(m<d)作为决策树 Trt 的特征集。

（4）根据算法(通常为无后剪枝的 CART)生成 Trt。

（5）按照 1)--4)生成大量决策树，构成随机森林(可并行)。

（6）预测样本时，通过投票法集成所有决策树的结果，进行预测。

Numpy 中提供了随机抽样函数，可实现有放回和无放回抽样。

numpy.random.choice(a, size=None, replace=True, p=None)，其中的各参数含义如表 3：

表 3 numpy 随机抽样函数参数表

参数	含义
a	设置抽样数据源。 1）如果是一维数组，就表示从这个一维数组中随机抽样； 2）如果是 int 型，就表示从 0 到 a-1 这个序列中随机抽样。
size	设置要抽样数据的维度。默认为 None，表示只抽样一个值。 1）可以为 int 型，如 m，表示要抽样 m 个数据； 2）也可以为 tuple，如(m, n, k)，则抽样的数量为 m*n*k 个，size 为(m, n, k)。

replace	设置是否为有放回抽样。 1）=True，则为有放回抽样； 2）=False，则为无放回抽样。
p	设置 a 中每个元素采样的概率。默认 None 表示 a 中每个元素被采样的概率相同。

程序示例：

（1）从[0,5)整数中有放回抽样 3 个数据：

```
>>> np.random.choice(5, 3)
array([0, 3, 4]) # random
>>> #This is equivalent to np.random.randint(0,5,3)
```

（2）从[0,5)整数中无放回抽样 3 个数据：

```
>>> np.random.choice(5, 3, replace=False)
array([3,1,0]) # random
>>> #This is equivalent to np.random.permutation(np.arange(5))[:3]
```

投票方式可以使用 Numpy 库中的 bincount 和 argmax 函数实现。

bincount 函数能先将 array 由小到大进行排序，然后对每个数值进行计数，并返回计数结果。需要注意的是，bincount 函数不能接受负数输入。

argmax 函数能够找到 array 中最大值，并返回最大值索引。

程序示例如下：

```
import numpy as np
# 分类的情况：输出7个弱评估器上的分类结果
r_clf = np.array([0, 2, 1, 1, 2, 1, 0])
```

```
b_result_clf = np.argmax(np.bincount(r_clf))
b_result_clf #集成算法在现在样本上输出的类别
```

如果是二分类，涉及到负类别的，可使用如下代码：

```
r_clf = np.array([1, 1, 1, -1, -1, -1, -1])
b_result_clf = 1 if (r_clf == 1).sum() > (r_clf == -1).sum() else -1
b_result_clf
```

如果投票的数量多个类别一致，无法按照少数服从多数进行判断时：

1）sklearn 使用的方法：随机返回一个类别（需要进行随机设置）。

2）自己编写代码，则返回编码数字更小的类别（如果使用 argmax 函数）。

5、RandomizedSearchCV(随机网格搜索)

GridSearchCV 是一种以穷举方式遍历参数空间内的所有离散值组合的超参数搜索方法，其缺点是搜索时间会随着数据量和超参数空间的增加而急剧增加，因而搜索空间的大小和训练所需的数据量多少是两个决定网格搜索运算速度的关键。RandomizedSearchCV(随机网格搜索)就是从缩小搜索空间的角度提升搜索速度的网格搜索策略，对参数空间内**按某种分布随机抽样**得到的子集进行搜

索。官网：https://scikit-learn.org/stable/modules/grid_search.html#randomized-parameter-search。随机网格搜索相较于 GridSearchCV 有两个优点：

（1）抽样得到的参数空间子集是可以独立于参数的数量和参数空间中的可能取值的；

（2）增加（不影响性能的）参数空间的数据量不会降低搜索效率。

具体来讲，就是当设置相同的全域参数空间时，随机网格搜索的运算速度会比枚举网格搜索更快；当设置相同的搜索次数时，随机网格搜索可以覆盖比枚举网格搜索更大的全域搜索空间；当随机搜索按照某种分布进行抽样时，在相同的参数区间内，抽取的参数个数相同，随机网格搜索可能得到与枚举网格搜索接近，甚至是超过枚举网格搜索的结果，如图 2 所示。

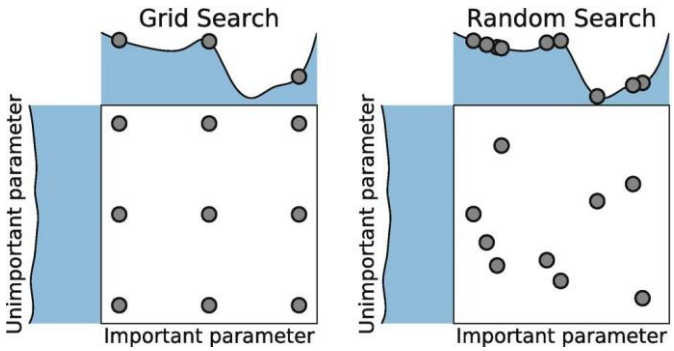

图 2 随机网格搜索按某种分布抽取参数效果示意图

sklearn 中 RandomizedSearchCV 的参数如表 4 所示，其中“param_distributions”、“n_iter”和“random_state”是 RandomizedSearchCV 不同于枚举网格搜索的参数。详见官网地址：https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV。

表 4　GridSearchCV 超参数表

参数	解释
estimator	调参对象，某评估器
param_distributions	**全域参数空间，可以是字典或者字典构成的列表，也可以是一种分布**
n_iter	**迭代次数，也是抽取的子空间的参数量**
scoring	评估指标，支持同时输出多个参数
n_jobs	设置工作时参与计算的 CPU 逻辑核数
refit	挑选评估指标和最佳参数，并在完整数据集上进行训练，得到一个泛化性能最好的模型
cv	交叉验证的折数
verbose	输出工作日志形式
pre_dispatch	多任务并行时任务划分数量
random_state	**随机数种子，控制抽样的随机性**
error_score	当网格搜索报错时返回结果，选择'raise'时将直接报错并中断训练过程，其他情况会显示警告信息后继续完成训练
return_train_score	在交叉验证中是否显示训练集中参数得分

●param_distributions：随机网格搜索中，除了可以设置搜索空间为离散值，还可以将搜索空间设置为某种连续分布。可以使用 scipy.stats 模块，其中包含多种采样参数分布，如指数分布 expon、

gamma 分布、连续均匀分布 uniform 或离散均匀分布 randint 等，sklearn 中的 utils.fixes 模块中提供了对数均匀分布 loguniform。

程序示例：

```
import scipy.stats as stats
from sklearn.utils.fixes import loguniform
```

```
# specify parameters and distributions to sample from
param_dist = {
    "average": [True, False],
    "l1_ratio": stats.uniform(0, 1),
    "alpha": loguniform(1e-2, 1e0),
}
```

示例中 uniform 是连续性均匀分布，默认为[0,1]之间的均匀分布，也可写成 stats.uniform(loc=0, scale=1)。参数 loc 和 scale 可以确定均匀分布的范围为[loc, loc+scale]。loguniform(1e-2, 1e0)表示[0.01,1]之间的对数分布。

上述程序示例取自官网“Comparing randomized search and grid search for hyperparameter estimation”实例，参见网址：https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html#sphx-glr-auto-examples-model-selection-plot-randomized-search-py。

RandomizedSearchCV 的属性与 GridSearchCV 完全相同，在此不再赘述。

6、HalvingSearchCV (对半网格搜索)

HalvingSearchCV 是同时调整搜索空间和参与训练样本数量的超参数搜索策略。官网：https://scikit-learn.org/stable/modules/grid_search.html#searching-for-optimal-parameters-with-successive-halving。

HalvingSearchCV 较前两种介绍过的超参数搜索策略更为复杂。它是一个迭代的搜索过程，第一次迭代时用少量的资源(resources)评估所有参数空间候选值，在下一次迭代时，只使用表现较好的一些参数，分配给更多的资源去评估，再从中选取表现好的参数，分配给更多的资源去评估，不断重复迭代过程。这里说的“资源”通常是训练样本的数量，也可以是其他任意数值参数，比如随机森林中的 n_estimators。

HalvingSearchCV 的提出基于以下考虑：假设现在存在数据集 D，从数据集 D 中随机抽样出一个子集 d，且“任意子集 d 的分布都与全数据集 D 的分布类似”。如果一组参数在整个数据集 D 上表现较差，那大概率这组参数在数据集的子集 d 上表现也不会太好。反之，如果一组参数在子集 d 上表现不好，我们也不会信任这组参数在全数据集 D 上的表现。如果“超参数在子集与全数据集上的表现一致”这一假设成立，那在网格搜索中，比起每次都使用全部数据来验证一组参数，或许可以考虑只带入训练数据的子集来对超参数进行筛选，这样可以极大地加速搜索过程。

当子集的分布越接近全数据集的分布，同一组参数在子集与全数据集上的表现越有可能一致。抽样子集越大，其分布越接近全数据集的分布，但是大子集会导致更长的训练时间，因此为了整体

训练效率，不可能无限地增大子集。HalvingSearchCV 算法设计了一个巧妙的流程，可以很好的权衡子集的大小与计算效率问题，具体的流程如下（“资源”为样本量）：

（1）首先从全数据集中无放回随机抽样出一个很小的子集 d_0（样本量为 S），在 d_0 上验证全部参数组合（参数空间中的候选参数数量为 C）的性能。根据 d_0 上的验证结果，淘汰评分排在后 1/2 的参数组合，保留下$\frac{1}{2}$C 个参数组合；

（2）然后，从全数据集中再无放回抽样出一个比 d_0 大一倍的子集 d_1（样本量为 2S），并在 d_1 上验证剩下的那一半参数组合的性能。根据 d_1 上的验证结果，再淘汰评分排在后 1/2 的参数组合，此时的参数组合数量为$\frac{1}{4}$C；

（3）再从全数据集中无放回抽样出一个比 d_1 大一倍的子集 d_2（样本量为 4S），并在 d_2 上验证剩下 1/4 的参数组合的性能。根据 d_2 上的验证结果，再淘汰评分排在后 1/2 的参数组合，此时的参数组合数量为$\frac{1}{8}$C，……，持续循环，如表 5 所示。

表 5　HalvingSearchCV 算法数据集和参数空间迭代表(factor=2)

迭代次数	子集样本量	参数组合数
1	S	C
2	2S	$\frac{1}{2}$C
3	4S	$\frac{1}{4}$C
4	8S	$\frac{1}{8}$C
…	…	（当 C 无法被除尽时，则向上取整）

在迭代过程中，用于验证参数的数据子集是越来越大的，而需要被验证的参数组合数量是越来越少的，最后留下的是在所有迭代中排名始终较高的超参数，HalvingSearchCV 的迭代效果如图 3 所示，示例程序见官网。

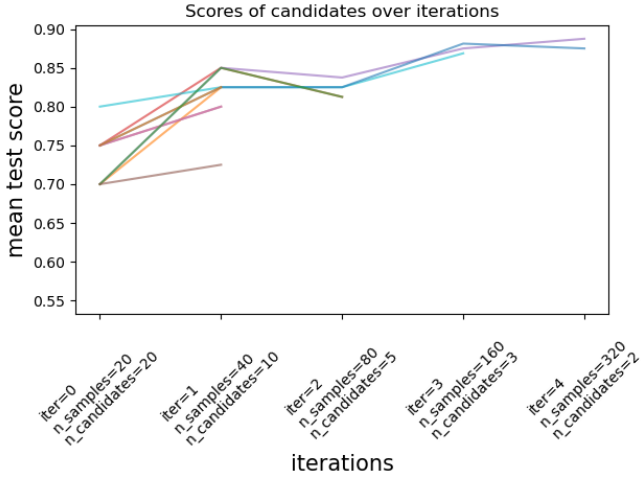

图 3 HalvingSearchCV 算法官网示例

迭代最终会因为所需的数据量超过最大的样本量，或者参数组合只剩 1 组而停止，可见迭代的次数是由初始样本集大小 d_0、数据量增加的倍数 factor、最大样本量和参数空间大小共同决定。

sklearn 中 HalvingSearchCV 的参数如表 6 所示，详见官网地址：https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingGridSearchCV.html#sklearn.model_selection.HalvingGridSearchCV。由于目前 sklearn 中，HalvingSearchCV 仍在实验阶段，后续可能会调整 API，因此要使用它，需要导入 enable_halving_search_cv。使用该模块需导入的模块示例如下：

```
>>> # explicitly require this experimental feature
>>> from sklearn.experimental import enable_halving_search_cv # noqa
>>> # now you can import normally from model_selection
>>> from sklearn.model_selection import HalvingGridSearchCV
```

表 6　HalvingSearchCV 超参数表

参数	解释
estimator	调参对象，某评估器
param_grid	参数空间，可以是字典或者字典构成的列表（参数空间是离散值）
factor	每轮迭代中新增的样本量的比例，同时也是每轮迭代保留的参数组合的比例
resource	设置每轮迭代中增加的验证资源的类型（默认为样本量）
max_resources	在一次迭代中，允许被用来验证任意参数组合的最大资源量
min_resources	首次迭代时，用于验证参数组合的资源量
aggressive_elimination	是否以全部数被使用完成作为停止搜索的指标，如果不是，则采取措施
cv	交叉验证的折数
scoring	评估指标，支持同时输出多个参数
refit	挑选评估指标和最佳参数，在完整数据集上进行训练
error_score	当网格搜索报错时返回结果，选择'raise'时将直接报错并中断训练过程 其他情况会显示警告信息后继续完成训练
return_train_score	在交叉验证中是否显示训练集中参数得分
random_state	控制随机抽样数据集的随机性
n_jobs	设置工作时参与计算的线程数
verbose	输出工作日志形式

●factor：每轮迭代中新增的样本量的比例，同时也是每轮迭代保留的参数组合的比例。例如，当 factor=2 时，下一轮迭代的样本量会是上一轮的 2 倍，每次迭代后保留 1/2 的参数组合。如果 factor=3 时，下一轮迭代的样本量会是上一轮的 3 倍，每次迭代后保留 1/3 的参数组合。虽然官网默认为 factor=3，但要依据实际样本量进行设置。

●resource：设置每轮迭代中增加的验证资源的类型，输入为字符串。默认是样本量，输入为"n_samples"，也可以是任意集成算法当中输入正整数的弱分类器，例如"n_estimators"或者"n_iteration"。

●max_resources：在一次迭代中，允许被用来验证任意参数组合的最大资源量，默认为最大样本量。当所需要的资源超过所能提供的最大资源量时，迭代停止。

●min_resources：首次迭代时，用于验证参数组合的资源量 r_0。可以输入正整数，或两种字符串"smallest","exhaust"。

当 resource="n_samples"时：

（1）输入正整数 n：表示首次迭代的样本量为 n。

（2）输入"smallest"：则根据规则计算 r_0：

1）对于回归类算法，r_0 = n_splits(交叉验证折数) * 2

2）对于分类算法，r_0 = n_classes(类别数量) * n_splits * 2

3）当资源类型不是样本量时，r_0=1。

（3）输入"exhaust"：将设置 r_0，以便最后一次迭代使用尽可能多的资源。即，最后一次迭代将使用小于 max_resources 的最大值，是 min_resources 和 factor 的倍数。目前，sklearn 中输入"exhaust"与输入"smallest"结果完全相同（此处是官网的 bug），因此，为了更好地利用数据，应该自己设置适合的 r_0 和 factor，示例程序如下：

```
factor = 2
n_samples = 20640
min_resources = 10
space = 4800
```

```
for i in range(100):
    if (min_resources*factor**i > n_samples) or (space/factor**i < 1):
        break

    print(i,
          "本轮迭代样本:{}".format(min_resources*factor**i),
          "本轮验证参数组合:{}".format(np.ceil(space/factor**i)))
```

```
0 本轮迭代样本:10 本轮验证参数组合:4800.0
1 本轮迭代样本:20 本轮验证参数组合:2400.0
2 本轮迭代样本:40 本轮验证参数组合:1200.0
3 本轮迭代样本:80 本轮验证参数组合:600.0
4 本轮迭代样本:160 本轮验证参数组合:300.0
5 本轮迭代样本:320 本轮验证参数组合:150.0
6 本轮迭代样本:640 本轮验证参数组合:75.0
7 本轮迭代样本:1280 本轮验证参数组合:38.0
8 本轮迭代样本:2560 本轮验证参数组合:19.0
9 本轮迭代样本:5120 本轮验证参数组合:10.0
10 本轮迭代样本:10240 本轮验证参数组合:5.0
11 本轮迭代样本:20480 本轮验证参数组合:3.0
```

●aggressive_elimination：bool，默认值=False。当数据总样本量较小，不足以支撑循环直到只剩下最后一组备选参数时，可以打开该参数。设置为 False 时，以全部样本被用完作为迭代结束的指标，这意味着最后一次迭代可能会评估多个候选参数。设置为 True 时，会重复使用首次迭代时的样本量，直到剩下的候选参数足以匹配样本量增加到最大时，只剩下最后一组备选参数。

HalvingSearchCV 由于实现与随机网格搜索和网格搜索不同，除了含有其他网格搜索的所有属性之外，还具有很多独有属性如表 7 所示。

表 7 HalvingSearchCV 的独有属性

参数	解释
n_resources_	每次迭代的资源数量（列表形式）
n_candidates_	每次迭代的候选参数的数量（列表形式）
n_remaining_candidates_	最后一次迭代之后剩下的候选参数的数量，其数值为： np.ceil(n_candidates[-1] / factor)

n_iterations_	实际迭代的次数 当 aggressive_elimination= True 时，n_iterations_= n_required_iterations_ 当 aggressive_elimination= False 时，n_iterations_= min(n_possible_iterations_, n_required_iterations_)
n_possible_iterations_	从 min_resources_开始迭代，直到样本量达到不超过 max_resources_的最大值而停止的迭代次数
n_required_iterations_	从 min_resources_开始迭代，直到参数候选数量=factor*1 而停止的迭代次数。当参数候选不足时，将小于 n_possible_ierations_。
max_resources_	最大资源数
min_resources_	首次迭代的最小资源数

这些主要的属性值，在调用 HalvingSearchCV 的 fit 方法时，会先计算出来，并显示（如下图所示），然后再进行参数搜索的迭代过程。

```
n_iterations: 12
n_required_iterations: 13
n_possible_iterations: 12
min_resources_: 10
max_resources_: 20640
aggressive_elimination: False
factor: 2
----------
iter: 0
n_candidates: 4800
n_resources: 10
Fitting 5 folds for each of 4800 candidates, totalling 24000 fits
```

四、实验内容

1、题目一：采用 scikit-learn 中的 RandomForestRegressor 对加利福尼亚房价数据集进行预测。

具体要求：

（1）导入数据集：加利福尼亚房价数据集是 sklearn 中自带的数据集，程序示例：

```
from sklearn.datasets import fetch_california_housing
```

```
X_california, y_california = fetch_california_housing(return_X_y=True)
```

通过查看数据量和维度、特征类型（离散 or 连续）、特征名、标签名、标签分布情况、数据集的描述等信息了解数据集。

（2）模型建立：分别使用 DecisionTreeRegressor 和 RandomForestRegressor 建立预测模型（参数默认即可）。

（3）模型评估：输出训练集和测试集评分（**以根均方误差 RMSE 为评估指标**）。

提示：cross_val_score 只能得到测试集分数且只允许使用一种评分指标，而 cross_validate 不仅可以使用多种评分指标(可通过 scoring 参数设置)，还可以以字典形式，返回每一折交叉验证分割上的训练时间(fit_time)、测试时间(score_time)、测试集分数(test_score)和训练集分数(train_score)。训练集分数(train_score)获得需要设置参数 return_train_score=Ture。程序示例如下：

```
randomforestRegressor =RandomForestRegressor()
valid_score = cross_validate(randomforestRegressor,X_california,y_california,cv=5
                             ,scoring = "r2"
                              ,return_train_score=True #是否返回训练集分数
                             )
```

```
valid_score
```

```
{'fit_time': array([13.24599981, 13.25500011, 13.17600012, 13.04399967, 12.96399999]),
 'score_time': array([0.10899997, 0.102     , 0.12399983, 0.08500028, 0.10200024]),
 'test_score': array([0.51402292, 0.70339333, 0.74220567, 0.61786497, 0.68136129]),
 'train_score': array([0.97412621, 0.97513643, 0.97489581, 0.97453262, 0.97564206])}
```

【讨论一】比较随机森林和决策树在数据集上的表现。

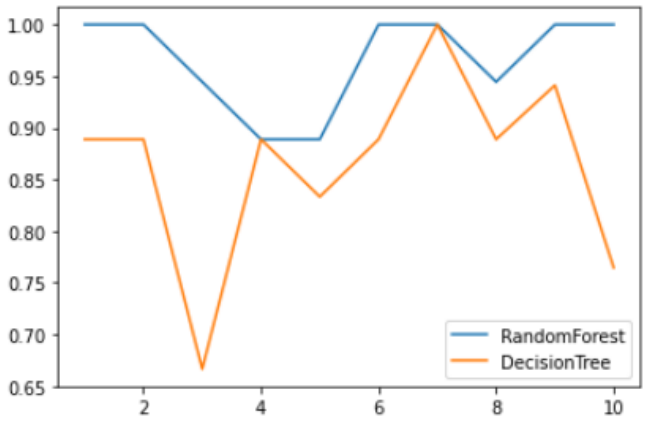

图 1 决策树与随机森林的 10 折交叉验证结果曲线比较（葡萄酒数据集）

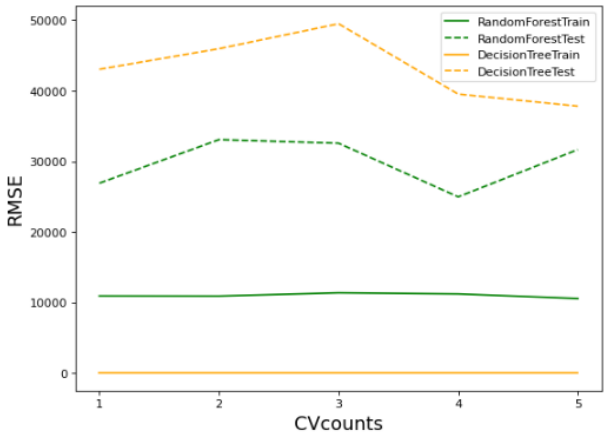

图 2 决策树与随机森林的 5 折交叉验证结果曲线比较（房价数据集）

提示：可将交叉验证评分进行可视化，如图 1、2 所示，示例分别展示了两种算法在解决分类和回归任务时，在默认参数情况下的交叉验证的模型评分。

【讨论二】随机森林中的 n_estimator 超参数如何选择？

提示：可采用学习曲线进行选择，如图 3、4 所示，学习曲线能够看到变化趋势，帮助确定超参数的搜索范围。

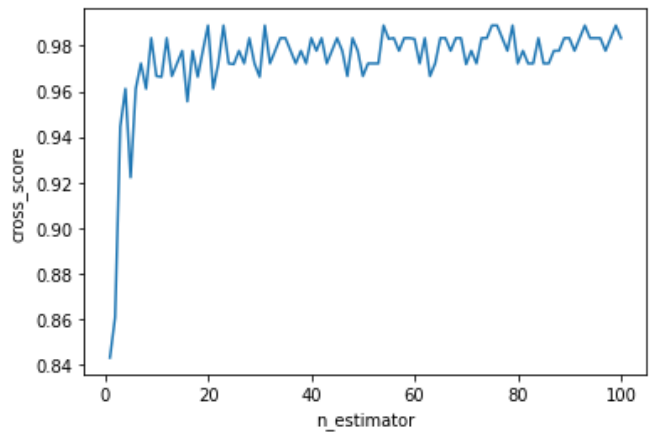

图 3 随机森林中基估计器(决策树)个数对模型准确率影响曲线比较（葡萄酒数据集）

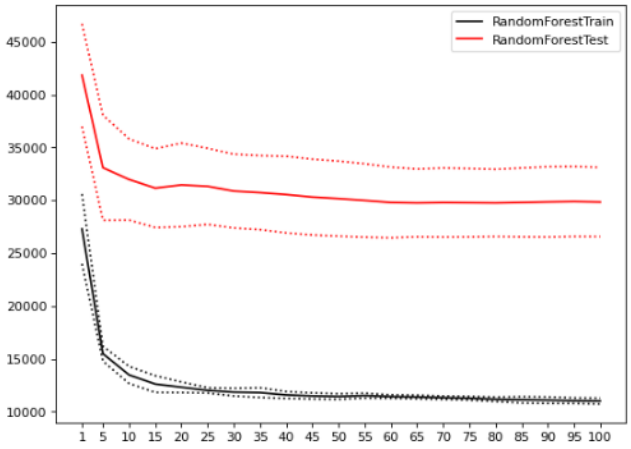

图 4 随机森林中基估计器(决策树)个数对模型均方误差影响曲线比较（房价数据集）

【讨论三】对于 RandomForestRegressor 模型，自行选择超参数搜索的方法，找到合适的超参数，最终将超参数在如下的交叉验证集上进行建模，并计算 RMSE 评分。介绍调参过程，并比较调参前后的效果。

交叉验证集的划分：`cv = KFold(n_splits=5, shuffle=True, random_state=1111)`

提示：超参数的搜索范围可根据建立的决策树的建树信息进行设定，决策树的建树信息可通过.tree_属性查看。示例如下：

```
reg_f = RandomForestRegressor(n_estimators=100, random_state=1000)
reg_f = reg_f.fit(X_california, y_california)
```

```
reg_f.estimators_[0].tree_.max_depth
```

```
35
```

2、题目二：编写随机森林算法，并对葡萄酒数据/加利福尼亚房价数据（只选择一种即可）进行预测，并展示模型评分，与 sklearn 自带的评估器建模结果进行对比。