随机森林实验

使用sklearn中的随机森林，实现预测label，并使用网格搜索调参。

RF框架

RF相比GBDT参数少，因为bagging中各个学习器之间没有依赖关系，同时减小了调参的难度。

RandomForestClassifier和RandomForestRegressor参数绝大部分相同

1) **n_estimators**: 也就是最大的弱学习器的个数。一般来说n_estimators太小，容易欠拟合，n_estimators太大，计算量会太大，并且n_estimators到一定的数量后，再增大n_estimators获得的模型提升会很小，所以一般选择一个适中的数值。默认是100。

2) **oob_score** :即是否采用袋外样本来评估模型的好坏。默认识False。个人推荐设置为True，因为袋外分数反应了一个模型拟合后的泛化能力。

3) **criterion:** 即CART树做划分时对特征的评价标准。分类模型和回归模型的损失函数是不一样的。分类RF对应的CART分类树默认是基尼系数gini,另一个可选择的标准是信息增益。回归RF对应的CART回归树默认是均方差mse，另一个可以选择的标准是绝对值差mae。一般来说选择默认的标准就已经很好的。

从上面可以看出， RF重要的框架参数比较少，主要需要关注的是 n_estimators，即RF最大的决策树个数。

决策树参数

它要调参的参数基本和GBDT相同，如下:

1) RF划分时考虑的最大特征数**max_features**: 可以使用很多种类型的值，默认是"auto",意味着划分时最多考虑N−−√N个特征；如果是"log2"意味着划分时最多考虑$log_2 N$个特征；如果是"sqrt"或者"auto"意味着划分时最多考虑$\sqrt{N}$个特征。如果是整数，代表考虑的特征绝对数。如果是浮点数，代表考虑特征百分比，即考虑（百分比xN）取整后的特征数。其中N为样本总特征数。一般我们用默认的"auto"就可以了，如果特征数非常多，我们可以灵活使用刚才描述的其他取值来控制划分时考虑的最大特征数，以控制决策树的生成时间。

2) 决策树最大深度**max_depth**: 默认可以不输入，如果不输入的话，决策树在建立子树的时候不会限制子树的深度。一般来说，数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值10-100之间。

3) 内部节点再划分所需最小样本数**min_samples_split**: 这个值限制了子树继续划分的条件，如果某节点的样本数少于min_samples_split，则不会继续再尝试选择最优特征来进行划分。 默认是2.如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

4) 叶子节点最少样本数**min_samples_leaf**: 这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 默认是1,可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

5）叶子节点最小的样本权重和**min_weight_fraction_leaf**：这个值限制了叶子节点所有样本权重和的最小值，如果小于这个值，则会和兄弟节点一起被剪枝。 默认是0，就是不考虑权重问题。一般来说，如果我们有较多样本有缺失值，或者分类树样本的分布类别偏差很大，就会引入样本权重，这时我们就要注意这个值了。

6) 最大叶子节点数**max_leaf_nodes**: 通过限制最大叶子节点数，可以防止过拟合，默认是"None”，即不限制最大的叶子节点数。如果加了限制，算法会建立在最大叶子节点数内最优的决策树。如果特征不多，可以不考虑这个值，但是如果特征分成多的话，可以加以限制，具体的值可以通过交叉验证得到。

7) 节点划分最小不纯度**min_impurity_split:** 这个值限制了决策树的增长，如果某节点的不纯度(基于基尼系数，均方差)小于这个阈值，则该节点不再生成子节点。即为叶子节点 。一般不推荐改动默认值1e-7。

上面决策树参数中最重要的包括最大特征数max_features， 最大深度max_depth， 内部节点再划分所需最小样本数min_samples_split和叶子节点最少样本数min_samples_leaf。

基础实现

参考GBDT的代码实现

```
import pydotplus
import pandas as pd
import numpy as np
from six import StringIO
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics, tree
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV

import matplotlib.pylab as plt

train = pd.read_csv('train_modified.csv')
target='Disbursed' # Disbursed的值就是二元分类的输出
IDcol = 'ID'
train['Disbursed'].value_counts()

x_columns = [x for x in train.columns if x not in [target, IDcol]]
X = train[x_columns]
y = train['Disbursed']

rf0 = RandomForestClassifier(oob_score=True, random_state=10)
rf0.fit(X,y)
print(rf0.oob_score_)
y_predprob = rf0.predict_proba(X)[:,1]
print("AUC Score (Train): %f" % metrics.roc_auc_score(y, y_predprob))
```

0.98005
AUC Score (Train): 0.999833

同样的数据集，相比GBDT高了很多。

网格搜索调参

对n_estimators

```
param_test1 = {'n_estimators':range(10,71,10)}
gsearch1 = GridSearchCV(estimator =
RandomForestClassifier(min_samples_split=100,

min_samples_leaf=20,max_depth=8,max_features='sqrt' ,random_state=10),
                        param_grid = param_test1, scoring='roc_auc',cv=5)
gsearch1.fit(X,y)
print(gsearch1.cv_results_, gsearch1.best_params_, gsearch1.best_score_)
```

{'n_estimators': 60},
0.8211334476626017)

得到了最佳迭代器迭代次数60

决策树最大深度max_depth和内部节点再划分所需最小样本数min_samples_split

```
param_test2 = {'max_depth':range(3,14,2), 'min_samples_split':range(50,201,20)}
gsearch2 = GridSearchCV(estimator = RandomForestClassifier(n_estimators= 60,
                                    min_samples_leaf=20,max_features='sqrt'
,oob_score=True, random_state=10),
   param_grid = param_test2, scoring='roc_auc',iid=False, cv=5)
gsearch2.fit(X,y)
print(gsearch2.cv_results_, gsearch2.best_params_, gsearch2.best_score_)
```

{'max_depth': 13, 'min_samples_split': 110},
0.8242016800050813)

划分所需最小样本数min_samples_split和叶子节点最少样本数min_samples_leaf

```
param_test3 = {'min_samples_split':range(80,150,20),
'min_samples_leaf':range(10,60,10)}
gsearch3 = GridSearchCV(estimator = RandomForestClassifier(n_estimators= 60,
max_depth=13,
                                    max_features='sqrt' ,oob_score=True,
random_state=10),
   param_grid = param_test3, scoring='roc_auc',iid=False, cv=5)
gsearch3.fit(X,y)
print(gsearch3.cv_results_, gsearch3.best_params_, gsearch3.best_score_)
```

{'min_samples_leaf': 20, 'min_samples_split': 120},
0.8248650279471544)

最大特征数max_features

```
param_test4 = {'max_features':range(3,11,2)}
gsearch4 = GridSearchCV(estimator = RandomForestClassifier(n_estimators= 60,
max_depth=13, min_samples_split=120,
                                    min_samples_leaf=20 ,oob_score=True,
random_state=10),
   param_grid = param_test4, scoring='roc_auc',iid=False, cv=5)
gsearch4.fit(X,y)
print(gsearch4.cv_results_, gsearch4.best_params_, gsearch4.best_score_)
```

{'max_features': 7},
0.8248650279471544)

重新测试

```
rf2 = RandomForestClassifier(n_estimators= 60, max_depth=13,
min_samples_split=120,
                                     min_samples_leaf=20,max_features=7
,oob_score=True, random_state=10)
rf2.fit(X,y)
print(rf2.oob_score_)
y_predprob = rf2.predict_proba(X)[:,1]
print("AUC Score (Train): %f" % metrics.roc_auc_score(y, y_predprob))
```

0.984
AUC Score (Train): 0.926915