# 随机森林

## 数据集

数据集（蛋白质数据集）来源：

<https://archive.ics.uci.edu/ml/datasets/HIV-1+protease+cleavage>

## 数据预处理

考虑到数据分类样本不均，处理方式为：过采样小样本，欠采样大样本。本次实验采用过采样小样本（采用直接复制小类样本，形成数量上的均衡），经统计如下：

|  |  |  |
| --- | --- | --- |
| 数据 | 正类（1） | 负类（0） |
| X1 | 402 | 344 |
| X2 | 375 | 1250 |
| X3 | 149 | 798 |
| X4 | 434 | 2838 |

处理后如下：

|  |  |  |
| --- | --- | --- |
| 数据 | 正类（1） | 负类（0） |
| X1 | 402 | 344 |
| X2 | 375\*3=1125 | 1250 |
| X3 | 149\*5=745 | 798 |
| X4 | 434\*6=2604 | 2838 |

## 实验过程

#### 3.1 调包实现

scikit-learn中和随机森林算法相关的类为RangeForestClassifier。

在scikit-learn中，RandomForest的分类类是RandomForestClassifier，回归类是RandomForestRegressor。

本次任务所用分类，如下：from sklearn.ensemble import RandomForestClassifier

1、类的构造函数为：

RandomForestClassifier(n\_estimators=10,criterion=’gini’,max\_depth=None,min\_samples\_split=2,min\_samples\_leaf=1,min\_weight\_fraction\_leaf=0.0,max\_features=’auto’,max\_leaf\_nodes=None,min\_impurity\_decrease=0.0,min\_impurity\_split=None,bootstrap=True,oob\_score=False, n\_jobs=1, class\_weight=None)

参数分析：

|  |  |
| --- | --- |
| 参数 | 参数说明 |
| n\_estimators | 森林中决策树的个数，默认是10 |
| Criterion | 采用何种方法度量分裂质量，信息熵或者基尼指数，默认是基尼指数 |
| max\_features | 寻求最佳分割时的考虑的特征数量，即特征数达到多大时进行分割。 |
| max\_depth | 树的最大深度 |
| min\_samples\_split | 分割内部节点所需的最少样本数量 |
| min\_samples\_leaf | 叶子节点上包含的样本最小值 |
| min\_weight\_fraction\_leaf | 能成为叶子节点的条件是：该节点对应的实例数和总样本数的比值，至少大于这个min\_weight\_fraction\_leaf值 |
| max\_leaf\_nodes | 最大叶子节点数，以最好的优先方式生成树，最好的节点被定义为杂质相对较少，即纯度较高的叶子节点 |
| min\_impurity\_split | 树增长停止的阀值。一个节点将会分裂，如果他的杂质度比这个阀值；如果比这个值低，就会成为一个叶子节点。 |
| min\_impurity\_decrease | 一个节点将会被分裂，如果分裂之后，杂质度的减少效果高于这个值。默认0. |
| Bootstrap | 是否采用有放回式的抽样方式，默认True |
| oob\_score | 是否使用袋外样本来估计该模型大概的准确率，默认False |
| n\_jobs | 默认1。拟合和预测过程中并行运用的作业数量。如果为-1，则作业数设置为处理器的core数。 |
| class\_weight  (dict, list or dicts, "balanced") | 如果没有给定这个值，那么所有类别都应该是权重1。  对于多分类问题，可以按照分类结果y的可能取值的顺序给出一个list或者dict值，用来指明各类的权重。  "balanced"模式，使用y值自动调整权重，该模式类别权重与输入数据中的类别频率成反比 |

该类主要方法为：

|  |  |
| --- | --- |
| 方法 | 描述 |
| apply(X) | 用构造好的森林中的树对数据集X进行预测，返回每棵树预测的叶子节点。所以结果应该是二维矩阵，  行为样本第几个样本，列为每棵树预测的叶子节点。 |
| decision\_path(X) | 返回森林中的决策路径 |
| fit(X, y[, sample\_weight]) | 用训练数据集(x, y)来构造森林 |
| get\_params([deep]) | 获得分类器的参数 |
| predict(X) | 预测X的类别 |
| predict\_log\_proba(X) | 预测X的类的对数概率，和predict\_proba类似，只是取了对数 |
| predict\_proba(X) | 预测X的类别的概率。输入样本的预测类别概率被计算为森林中树的平均预测类别概率。单个树的类概率是叶中同一类的样本的比率。因为叶子节点并不是完全纯净的，它也有杂质，不同种类所占恶比率是不一样的，但肯定有一类纯度很高。返回值是array of shape = [n\_samples, n\_classes] |
| score(X, y[,sample\_weight]) | 返回给定的数据集（数据集指定了类别）的预测准确度 |
| set\_params(\*\*params) | 设置决策树的参数 |

本次实验选择的参数为：

n\_estimators=10,max\_depth=10,class\_weight="balanced"

实验结果（准确率）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Train  Test | X1 | X2 | X3 | X4 |
| X1 | 0.95 | 0.92 | 0.79 | 0.71 |
| X2 | 0.91 | 0.95 | 0.79 | 0.72 |
| X3 | 0.68 | 0.61 | 0.94 | 0.76 |
| X4 | 0.67 | 0.60 | 0.78 | 0.91 |

将max\_depth=10修改为20，准确率提升了

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Train  Test | X1 | X2 | X3 | X4 |
| X1 | 0.99 | 0.97 | 0.72 | 0.75 |
| X2 | 0.93 | 1 | 0.73 | 0.77 |
| X3 | 0.70 | 0.62 | 1 | 0.74 |
| X4 | 0.70 | 0.60 | 0.71 | 0.98 |

继续增大max\_ depth发现准确率并没有提升（反而有所下降，应该是过拟合了），所以选择最优的参数为20。

将n\_estimators=100测试性能如下，准确率有所提高。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Train  Test | X1 | X2 | X3 | X4 |
| X1 | 1 | 0.97 | 0.69 | 0.75 |
| X2 | 0.93 | 1 | 0.72 | 0.77 |
| X3 | 0.72 | 0.61 | 1 | 0.75 |
| X4 | 0.73 | 0.58 | 0.76 | 0.98 |

#### 3.2 numpy实现

在本次实验中，所用tree直接选择生成上次的CART树。

1、确定采样方式。

采用随机有放回的采样，对于数据集大小为N的样本，随机采样N次构建数据集的子集，如下：

def bootstrap\_sample(X,Y):

N,M=X.shape

idxs=np.random.choice(N,N,replace=True)

return X[idxs],Y[idxs]

2、构建随机森林类。

RandomForest(n\_trees,max\_depth,n\_feats,classifier=True,criterion="entropy")

|  |  |
| --- | --- |
| 参数 | 说明 |
| n\_trees | 集成中要使用的决策树的数量 |
| max\_depth | 停止生长的每个决策树深度。如果为None,就直到叶子结点变纯为止。 |
| n\_feats | 每次分裂要采样的特征的数量。 |
| Classifier | 是否是分类任务。默认为True |
| Criterion | 分裂标准，默认是”entropy”，可选择用”gini”,”mse”等。 |

该类所定义的函数

|  |  |
| --- | --- |
| 函数名 | 说明 |
| fit(X,Y) | 根据n\_trees和bootstrap采样方式来生成每个决策树，并拟合每棵决策树，存入数组trees=[] |
| predict(X) | 预测数据集X中的每个样本的类别值（对数据集X中的每个样本x，对所有的trees中的tree都进行一次分类，并存入数组中） |
| \_vote(predictions) | 采用相对多数投票法，至少能输出一个预测值。 |

函数大致实现思路：

|  |  |
| --- | --- |
| 函数名 | 函数大致实现思路 |
| fit(X,Y) | 循环n\_trees次，每次都随机采样(X,Y),并实例化DecisionTree,以及fit此棵决策树。最后将此棵决策树对象放入trees数组中。 |
| predict(X) | 遍历trees数组中的每棵决策树，对X中的每个样本进行分类，将类别标签存入数组中。这样数组中就有n个类别值。（n个决策树） |
| \_vote(predictions) | 通过n棵决策树的预测值数组，输出预测类别最多的那个值作为最终预测值。 |

实验结果如下：

RandomForest(n\_trees=10,max\_depth=20,criterion="gini",n\_feats=None)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Train  Test | X1 | X2 | X3 | X4 |
| X1 | 0.88 | 0.85 | 0.72 | 0.84 |
| X2 | 0.85 | 0.87 | 0.70 | 0.79 |
| X3 | 0.72 | 0.65 | 0.83 | 0.76 |
| X4 | 0.73 | 0.65 | 0.77 | 0.78 |

单纯用之前写的CART决策树进行分类的结果（并没有定义max\_depth，使之不纯即分裂，导致训练集过拟合，与上面相比，从预测的角度说明，整体效果不如集成）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Train  Test | X1 | X2 | X3 | X4 |
| X1 | 1 | 0.96 | 0.61 | 0.71 |
| X2 | 0.91 | 1 | 0.65 | 0.74 |
| X3 | 0.71 | 0.65 | 1 | 0.66 |
| X4 | 0.70 | 0.65 | 0.67 | 1 |

dt=DecisionTree(criterion="gini",max\_depth=20, n\_feats=None)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Train  Test | X1 | X2 | X3 | X4 |
| X1 | 0.88 | 0.84 | 0.69 | 0.83 |
| X2 | 0.84 | 0.87 | 0.69 | 0.78 |
| X3 | 0.72 | 0.65 | 0.81 | 0.76 |
| X4 | 0.73 | 0.65 | 0.75 | 0.78 |

效果略低于集成（百分之2到百分之5左右），说明了集成的好处。