**数据分析与处理部分实践作业 5#**

# 随机森林算法

## 随机森林算法介绍

### 1.1.1 Bagging

并行式集成学习方法，采用自助采样法经过m次随机采样操作，得到m个样本的采样集，初始训练集中约有63.2%的样本出现在样本集中。

采样出T个含m个训练样本的采样集，基于每个样本集训练出一个基学习器，再将基学习器结合。

Bagging算法过程：

对1...T个样本集，

输出：

此外，自助采样过程给Bagging带来一个优点：基学习器只用到训练集中63.2%的样本，剩下约36.8%的样本可用作验证集来对泛化性能进行“包外估计”。

包外样本其他作用：基学习器是决策树，可使用包外样本来辅助剪枝；当基学习器是神经网络时，可用包外样本来辅助早期停止以减小过拟合风险。

### 1.1.2 Random Forest

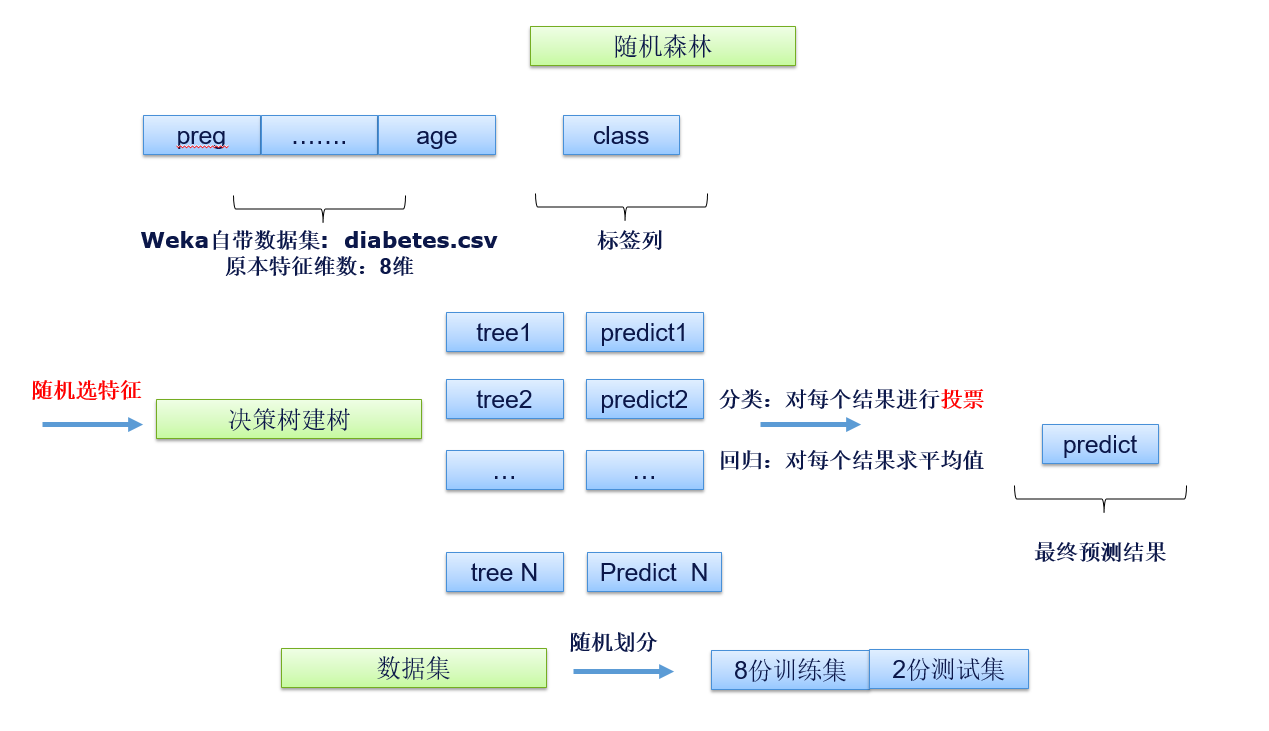
随机森林，简称：RF，是Bagging的扩展变体。RF在以决策树为基学习器的Bagging集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。

在RF中，对基决策树的每个结点，从该结点的属性集合中随机选择一个包含k个属性的子集，再从子集中选择一个最优属性用于划分，推荐值。

随机森林中的基学习器多样性来自于样本扰动和属性扰动，加入属性扰动会使个体学习器的性能有所下降，随着学习器的数目增加，随机森林可收敛到更低的泛化误差。

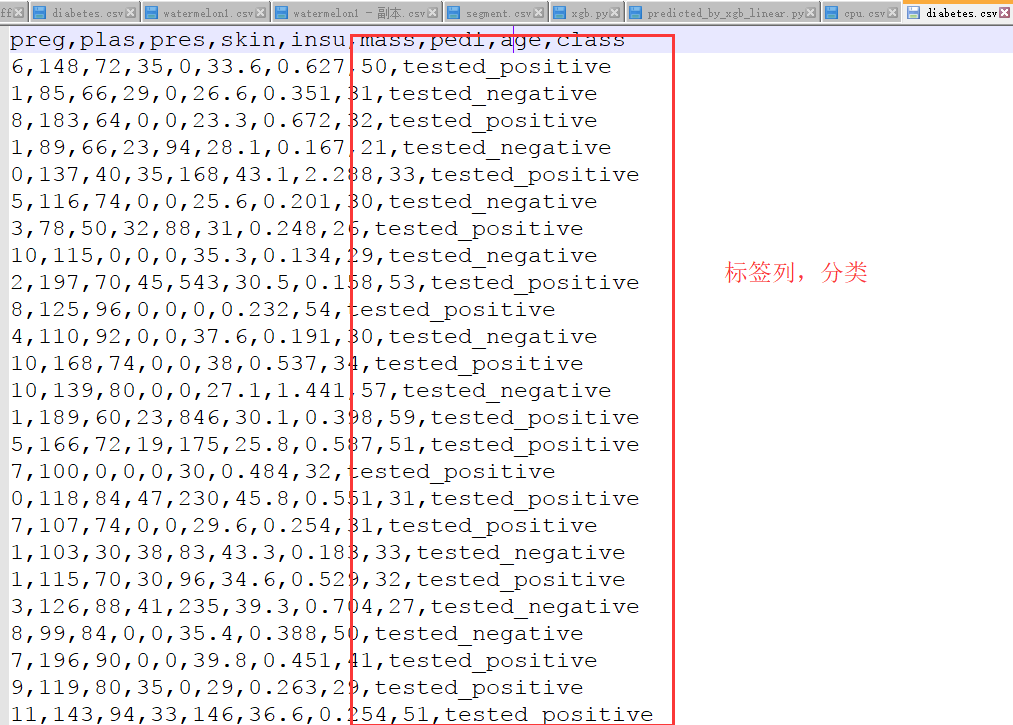
## 随机森林算法实现

思路流程图：

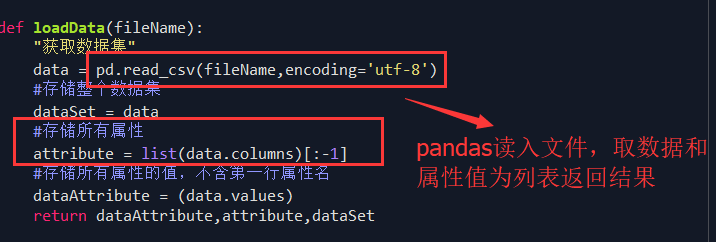


### 1.2.1 传入数据

1、数据格式，逗号分割的csv文件，一共8维特征，class为标签列：



2、函数构造：



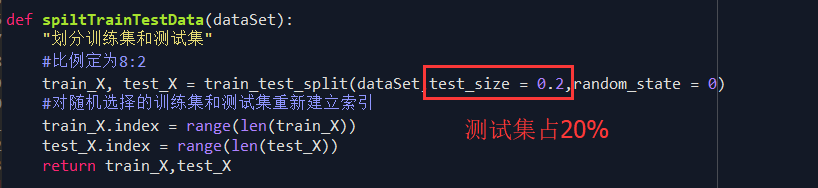
3、读入csv文件，返回的数据集和属性列表。

4、数据读入结果：



### 1.2.2 按比例随机划分数据集

1、函数构造：



2、函数思路：

（1）调用sklearn.cross\_validation 库中的train\_test\_split函数进行训练集和测试集划分；

（2）传入读取的数据集，设置测试集的比例为20%，返回的就是切分好的训练集和测试集；

（3）切分后dataframe类型的索引是乱序，因为重新建立索引，返回切分的数据集。

3、数据集随机切分结果：

（1）训练集情况：

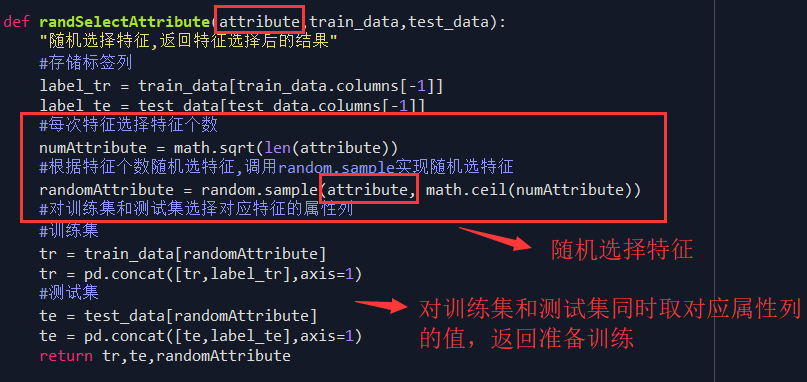


（2）测试集情况：



### 1.2.3 随机划分属性

1、构造函数：



2、思路：

（1）传入属性列、训练集、测试集；

（2）调用随机sample函数随机取特征；

（3）根据选出来的特征，截取训练集和测试集，返回结果。

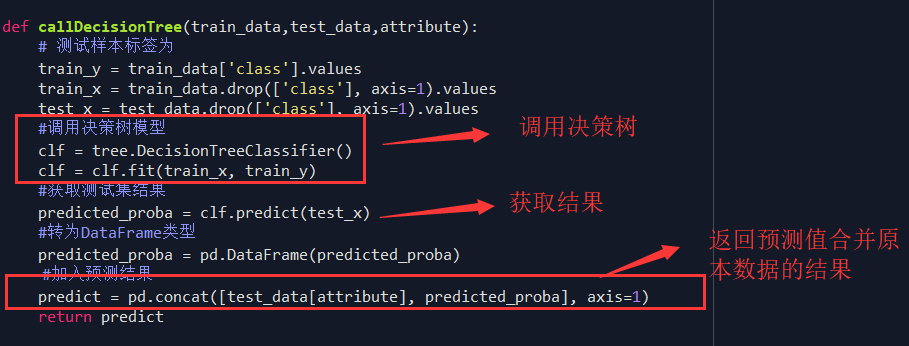
3、对于每棵树随机选特征的结果：



可以发现，随机森林的每棵训练树的特征的随机选择的。

### 1.2.4 随机选特征后的数据集跑决策树模型

1、构造函数：



2、思路：

（1）传入训练集和测试集；

（2）训练决策树模型，测试集得到预测结果；

（3）预测值合并之前的属性列返回结果。

### 1.2.5 对每棵的预测结果进行投票

1、函数构造：



2、思路：

（1）传入需要训练决策树的棵树，以及训练集和测试集；

（2）set取集合，取class一共有哪些类别，有一种类别就在数据后面加这些列，用于统计分到该类别的次数；

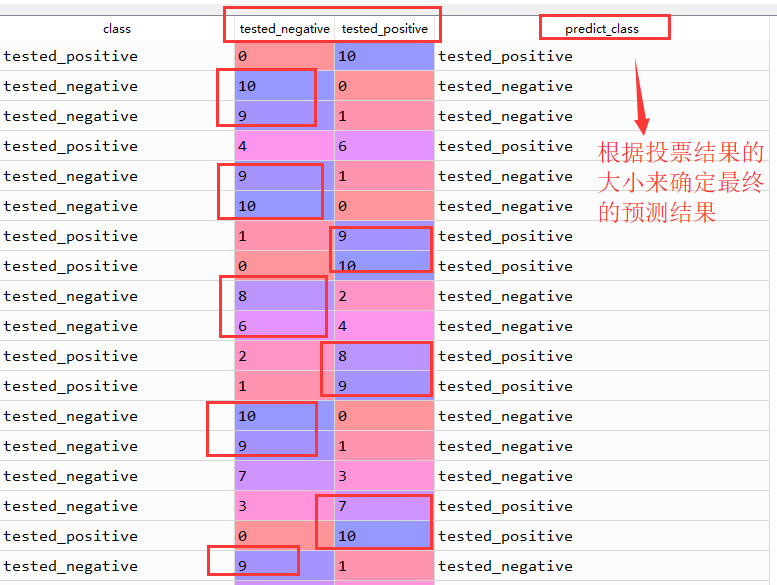
（3）循环所有棵树，循环一次建一棵树，建树时要随机选特征，返回的训练集和测试集才作为训练决策树的数据集；

（4）统计票数，对于当前这棵树的预测结果，对它每个样本属于哪类进行计数，比如属于0类，就在0列对应样本的位置加1；

（5）投票操作，对于之前统计的票数，对于一个样本，票数的最大值对应的字段就是它的类别，也就是投票最终结果，就是分类的最终结果。

不足：时间复杂度相对较高，导致投票过程比较费时间。做分类比回归要复杂一些，投票过程相比于回归要复杂，回归只要将每棵树的结果进行取均值即可，而分类的投票相对复杂很多。

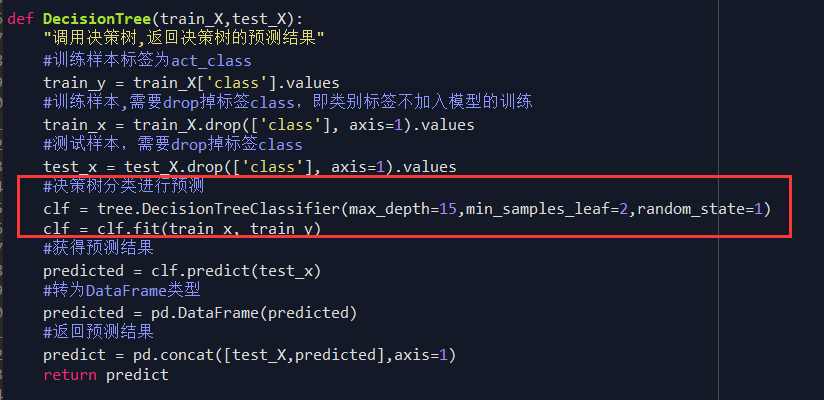
3、查看投票结果：



本次随机森林训练了10课树，例如第一条数据样本，tested\_positvie票数为9，而tested\_negative票数为1，那么投票结果就是tested\_positive票选成功，那么预测这条样本的结果就是tested\_positive。对所有样本执行的操作均相同。

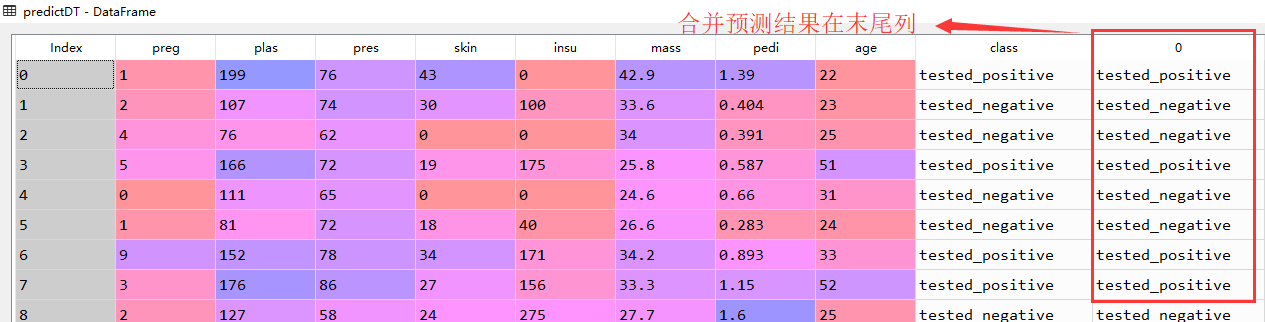
### 1.2.6 对数据集进行决策树训练

1、函数构造：



2、为了验证随机森林和决策树的效果，所以这里传入数据集进行决策树训练，返回结果等待计算F1值。

3、调用决策树的预测结果：



将预测结果concat到原本数据集，目的是为了方便后面调用评价函数进行计算。

### 1.2.7 F1值计算

1、函数构造：



2、思路：

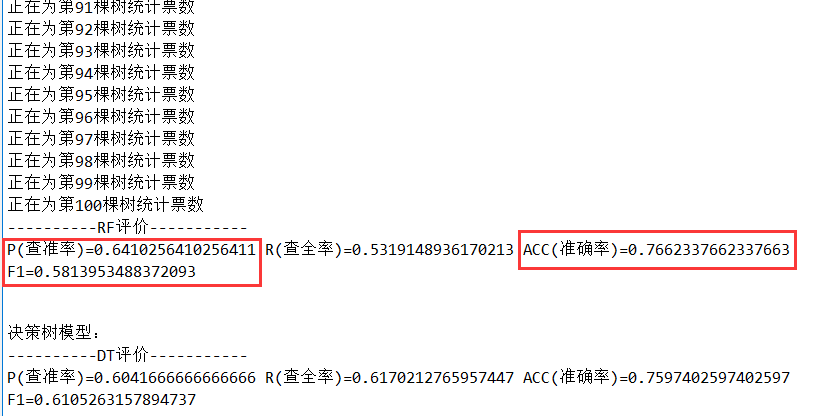
（1）根据预测值和真实值分别计算TP/FN/FP/TN值；（2）然后根据公式来计算分类效果；（3）打印评分结果。

### 1.2.8 调用函数得到结果

1、调用函数：

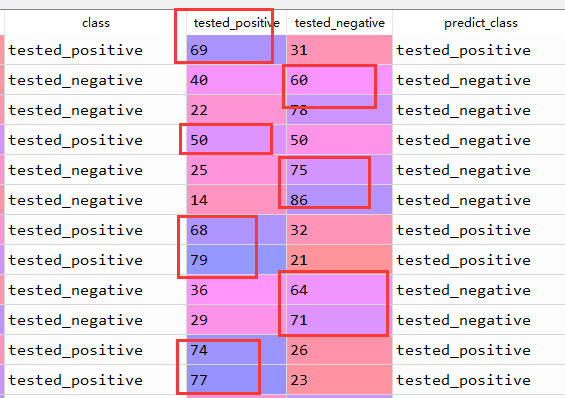


2、特征选择sqrt(特征维数)，随机森林建树100棵的控制台结果：



小结：由于数据集本身效果不好，调用模型评分不高。其次在跑随机森林模型的时候，数据维数一共8维，如果每次取sqrt(8)即3维的话，特征维数太少，导致随机性太强，效果不是很明显。如果维数增加一些，会有提升。

查看投票结果：

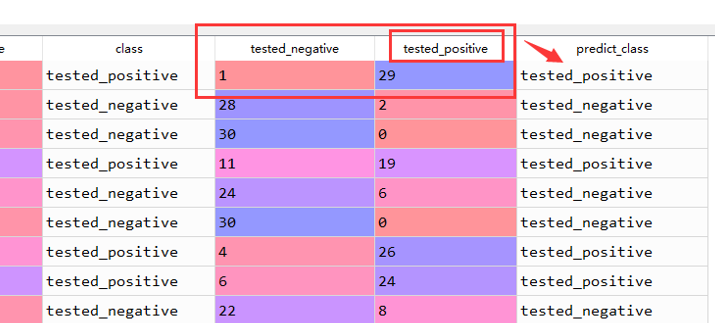


3、特征选择sqrt(特征维数)+3，随机森林建树30棵的控制台结果：



小结：由于特征维数太少，所以把特征维数增加一些，哪怕是树的棵树减少，也会提升比较明显。

查询投票结果：



例如第一条数据样本，tested\_positive的票数为29票，那么就预测该样本为tested\_positive。

### 1.2.9 小结

随机森林实现比较简单，两个关键步骤：（1）建每棵树时随机选特征；（2）对决策树结果进行投票操作。本文由于属性数较少，所以随机选特征后，提升并不明显，特征覆盖多一点儿效果会好一些。其次是投票算法实现的时候时间性能上比较低，导致跑算法比较耗时间。

对于随机森林回归的算法实现更为简单，调用决策树回归，随机选特征，对每棵树的预测结果取平均值即为最终结果。

# Stacking算法

## Stacking算法介绍

Stacking是学习法的代表，通常把个体学习器称为初级学习器，用于结合的学习器称为次级学习器或者元学习器。Stacking先从初始数据集训练出初级学习器，然后“生成”一个新数据集用于训练次学习器。在新数据集中，初级学习器的输出被当做样例输入特征，而初始样本的标记仍被当做样例标记。初级集成是异质的。

stacking 产生方法是一种截然不同的组合多个模型的方法，它讲的是组合学习器的概念，但是使用的相对于bagging和boosting较少，它不像bagging和boosting，而是组合不同的模型，具体的过程如下：

1.划分训练数据集为两个不相交的集合。

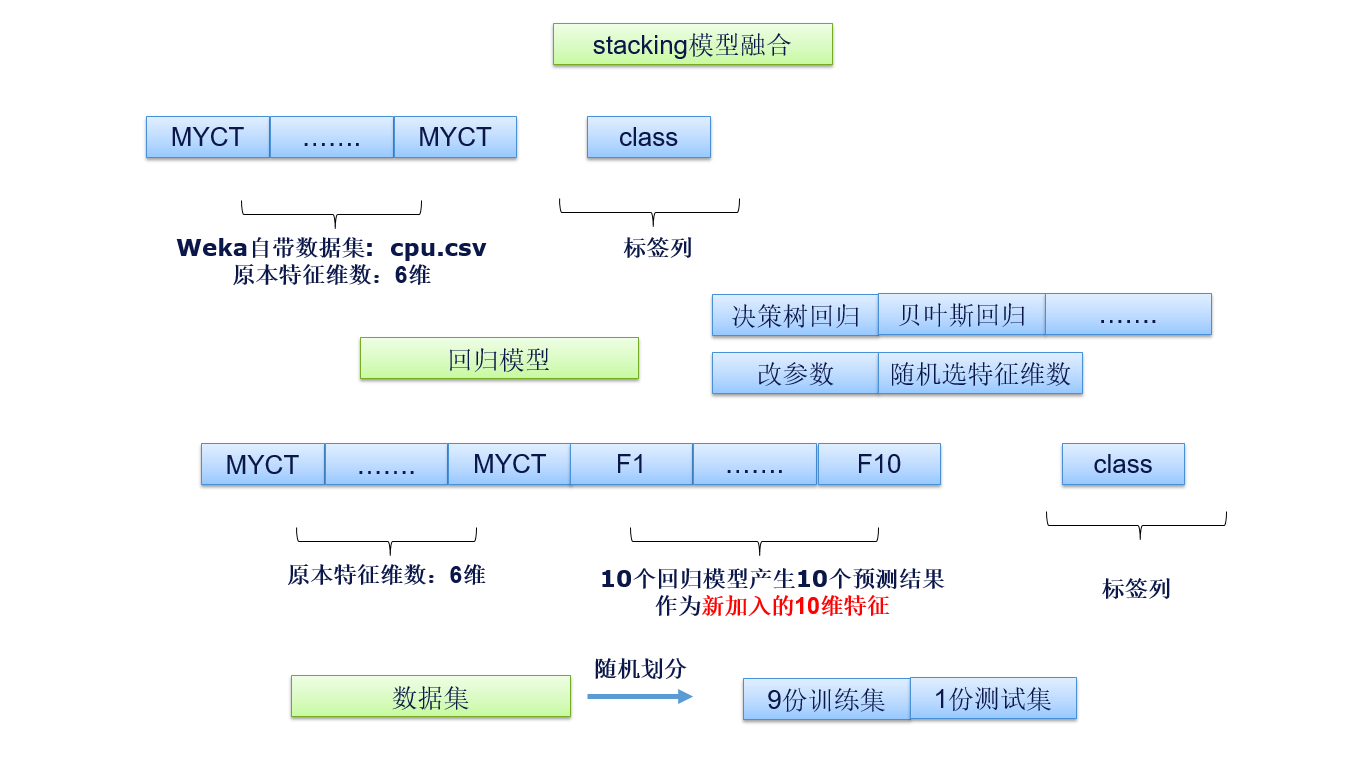
2. 在第一个集合上训练多个学习器。

3. 在第二个集合上测试这几个学习器

4. 把第三步得到的预测结果作为输入，把正确的回应作为输出，训练一个高层学习器。

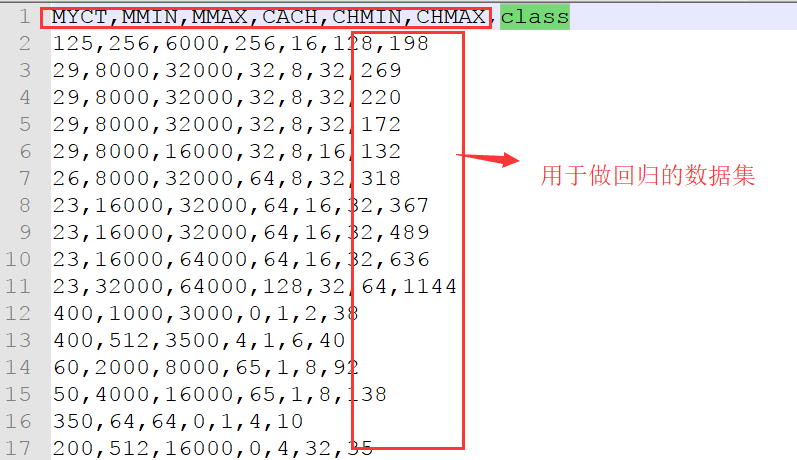
## Stacking算法实现

流程图：

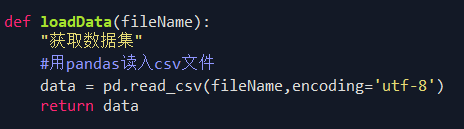


### 2.2.1 传入数据

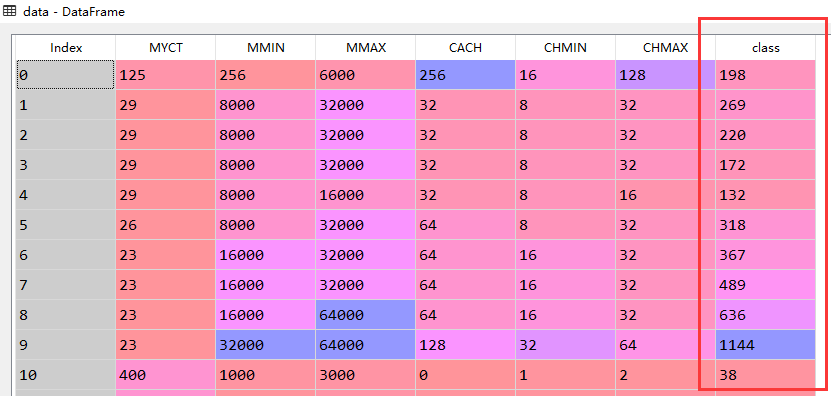
1、数据选择weka中做自带做回归的数据集：cpu.csv数据格式，逗号分割的csv文件，一共6维特征，class为标签列



2、函数构造：

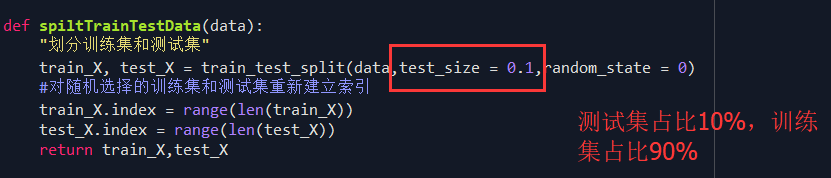


3、利用pandas读入csv文件，读入数据集结果：



### 2.2.2 按比例随机划分数据集

1、函数构造：



2、函数思路：

（1）调用sklearn.cross\_validation 库中的train\_test\_split函数进行训练集和测试集划分；

（2）传入读取的数据集，设置测试集的比例为10%，返回的就是切分好的训练集和测试集；

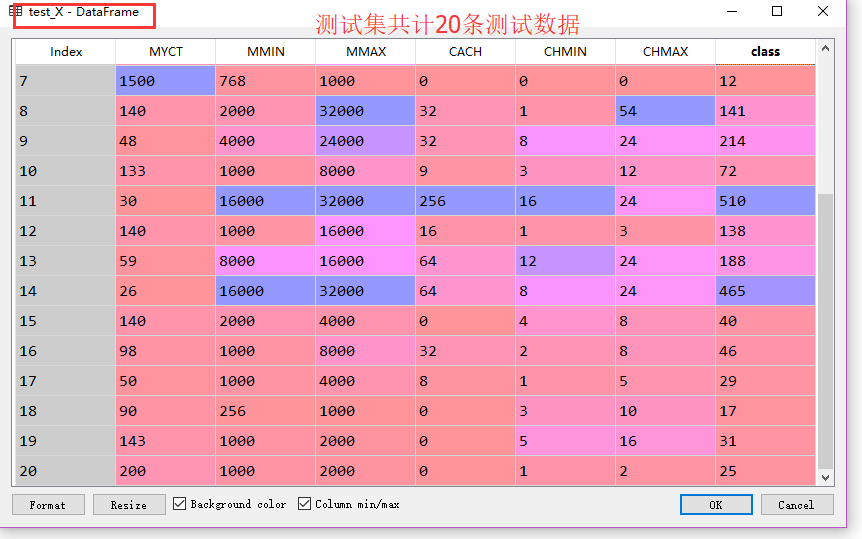
（3）切分后dataframe类型的索引是乱序，因为重新建立索引，返回切分的数据集。

3、随机切分的训练集和测试集情况：

（1）训练集情况：



（2）测试集情况：



### 2.2.3 stacking中第一层模型实现

#### 2.2.3.1 更改模型参数进行训练

1、函数构造：



上面是基回归算法使用决策树回归的代码，本文使用了的基模型算法有：

（1）决策树回归

（2）Ridge回归

（3）背景梯度提升回归

（4）贝叶斯回归

（5）随机森林回归

对于每个模型更改了两个模型参数，返回10个回归预测结果，作为新增的10维特征合并到原始的数据样本中。为了增强基学习器的不同性，还使用了随机选特征，在随机森林回归中，每次只随机选80%的特征数进行训练，2.2.3.2中重点介绍。

#### 2.2.3.2 随机选择数据集特征子集进行训练

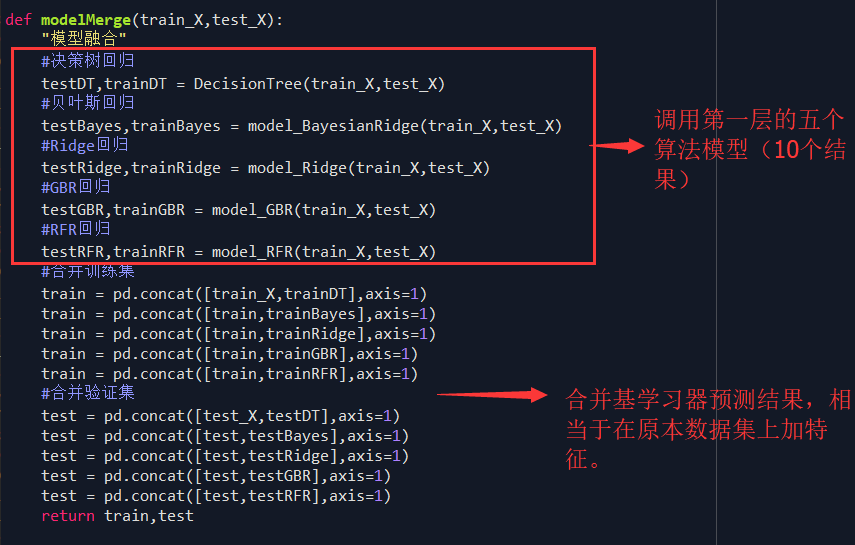
1、函数构造：



主要为了初级学习器多样性，引入了随机选特征这一环节，让stacking算法第一层出来的预测结果进行具有多样性。

### 2.2.4 合并基学习器结果

1、函数构造：



2、思路：

分别跑第一层的模型，将返回的预测结果当做新的特征，将新的特征concat到原本的数据集上面，最终返回的结果作为第二层线性回归的输入。

### 2.2.5 次学习器

1、函数构造：

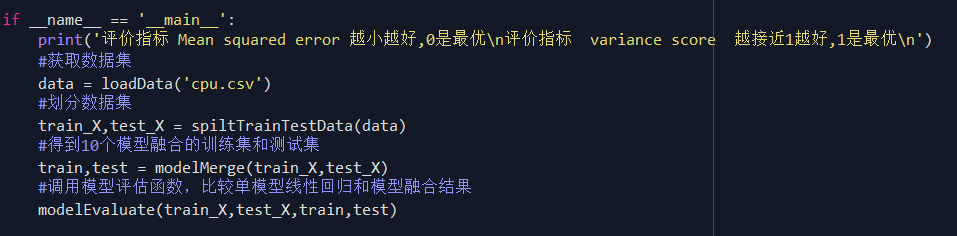


2、思路:

对于原始数据集和初级学习器分别进行训练线性回归算法，对比单模型线性回归算法和stacking模型融合的效果。

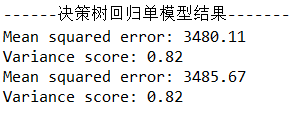
### 2.2.6调用函数得到结果

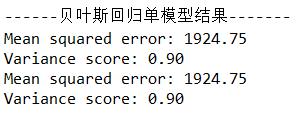
1、调用函数：

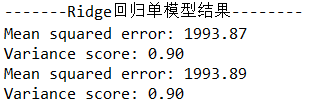


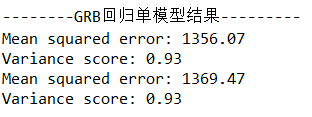
2、结果展示：

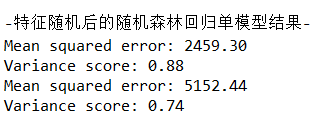


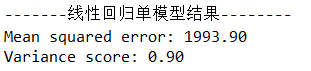


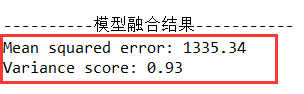












评价指标：Mean squared error，越小越好,0是最优；评价指标：variance score，越接近1越好,1是最优。根据结果可以看出由stacking模型融合后效果较初级学习器有提升，效果并不是特别好也不算特别差。

### 2.2.7小结

本次实现stacking模型融合在初级学习器上主要使用了算法模型的不同、模型的参数的不同、以及数据集的特征数的不同来增加初级学习器的多样性。因为是搞的回归，次级学习器使用的算法是线性回归算法，模型的评价使用了sklearn库中的均方误差和误差评分来评价模型。Stacking算法很容易过拟合，以后能不用最好还是不要用。